ROYAL INSTITUTE
OF TECHNOLOGY

NFC-Enable System Design in Wireless Sensor Network

YIN HUA

Master's Thesis at iPack Center, KTH
Supervisor: Qiang Chen, Zhibo Pang
Examiner: Prof. Lirong Zheng

January 2013

Abstract

Wireless Sensor Network (WSN) have the potential to greatly affect every part of industrial and people's lifestyle. For this reason, iPack VINN Excellence Center contributes to wireless tracking platform for fresh food and lifestyle. Every new idea or technology is attempted to integrate to the WSN for more efficient, better user experience and lower power consumption. Meanwhile Near Field Communication (NFC), a short-range wireless connectivity technology, which can make communication easily,safety and intuitively arousing iPack interest. So this master thesis focus on integrating NFC technology into existing systems to build a NFC-enable Wireless Sensor Network system. And with this system, only one simple touch, data from sensor node can be transmitted to mobile phone or tablet. Furthermore parameters of sensor node also can be configured easily by using above devices. So basically the NFC peer-to-peer communication protocol is mainly used. To implement and test the functions of the demonstration, a sets of hardware is needed to chosen and bought.

How to design the system without changing old WSN is very tricky. To design a NFC adapter which can connect existing WSN with NFC part is the solution of this thesis. So the main task was designing a NFC adapter which could be connected with either mobile phone/tablet or sensor node. It was the NFC adapter that makes mobile phone/tablet or sensor node NFC enabled. For the connection method, the high speed UART interface was chosen to connect with sensor node. The architecture of NFC adapter includes two main parts, A NFC chip (PN532) from NXP and a MCU(VNC2) from FTDI. The PN532 uses its antenna to send or receive data with different NFC protocols. The VNC2 is used to store sensor collected data and sends command though UART to control the PN532.

Learning to use the PN532 was a tough task during the thesis work. Both official manuals and demo application are helpful for understanding the PN532 controlling. In addition, We analysed the sniffer data from demo application and code from NXP software design kit (SDK). which helped us to know the process of the PN532 peer-to-peer communication. After learning from official application, user manual and monitoring software/hardware design kid applications, we began to design our own hardware suitable SDK for the NFC adapter. At first we connected the hardware parts. When hardware connection was ready, we wrote and tested the firmware for VNC2 platform. Then due to the reason that Windows is more stable than our build VNC2 platform system at that moment, we wrote our own software design kid for NFC adapter under Windows OS first. The basic idea of software design kid is easy to use, modify and integrate into any other platforms. At the end of the thesis project, we integrated our own SDK into VNC2. When integration was done, a lot of stability and performance validation were done. Based on the result of testing, we optimized and modified our SDK and tested it again.

This thesis project basically handles out a new ideal of integrating NFC to existing wireless sensor network to make WSN NFC enable. To prove the idea, we made a demo to show the enhanced sensor node and the results are satisfied. However there still has a lot of works and a lot of improvement should can be done in the future.
Keywords:Near Field Communication; Wireless Sensor Network; peer-to-peer; NFC enable; PN532; VNC2.

Contents

Contents **iv**

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Outline of the Thesis Report . . . 2

I Background Study **3**

2 Basic Knowledge **5**
- 2.1 NFC Technology Roadmap . . . 5
  - 2.1.1 ISO 14443-A/B, FeliCa . . . 5
  - 2.1.2 ISO/IEC 18092 or ECMA-340 . . . 5
  - 2.1.3 Working Mode . . . 6
- 2.2 The Comparative on wireless transmission . . . 7
  - 2.2.1 Bluetooth2.1 . . . 7
  - 2.2.2 ZigBee . . . 8
  - 2.2.3 802.11(WiFi) . . . 8
  - 2.2.4 IrDa . . . 8

3 Project Environment **9**
- 3.1 Hardware Environment . . . 9
  - 3.1.1 Tablet with NFC Chip . . . 9
  - 3.1.2 NFC Adapter . . . 9
  - 3.1.3 PN532 features and benefits . . . 13
  - 3.1.4 PN532 Demo Board . . . 14
  - 3.1.5 Galaxy Tab . . . 15
  - 3.1.6 slave node(MSP430) . . . 16
  - 3.1.7 FTDI VNC . . . 17
- 3.2 Software Environment . . . 18
  - 3.2.1 PC RS232 COM port programme IDE . . . 18
  - 3.2.2 Serial port Monitor and Sniffer Software . . . 19
  - 3.2.3 VNC2 IDE . . . 20

II NFC Adapter Implementation 21

4 Four Ways Implementation Trade Off 23
4.1 Libnfc 23
4.2 NFC-FRI SDK integrate 23
4.3 Porting NFC HAL Software 24
4.4 Sniffer and Write own SDK 26

5 How to Use The PN532 Command 29
5.1 PN532 host link protocol 29
5.1.1 Normal and Extended Information Frame 30
5.1.2 ACK Frame and NACK Frame 30
5.1.3 Error Frame 31

6 Implement Application on VNC 33
6.1 Connection of Hardware 33
6.1.1 Additional Lines–IRQ(DTR) and HREQ(DSR) 34
6.2 Application Code 35
6.2.1 Low Power mode and Wake up 35
6.2.2 Baud Rate Setting 37
6.2.3 As a NFC Target setting 37
6.2.4 As a NFC Initiator setting 37
6.2.5 Data Exchange 39
6.2.6 Overview of our own developing NXP PN532 SDK 41
6.3 SDK Integrated Into VNC 41
6.3.1 VNC firmware code 41
6.3.2 Integrated 44
6.3.3 testing and optimize 46
6.3.4 Final Demonstration 47

IIIConclusion and Future Work 49

7 Conclusion 51
7.1 Overview of The Thesis Work 51

8 Future Work 53
8.1 Support SPI and I2C Interface 53
8.2 Improve and Optimize SDK Code 53
8.3 communicate directly with NFC-enable tablet 53

Reference 55

Chapter 1

Introduction

Wireless sensor network (WSN) consists of spatially distribute autonomous sensors to monitor physical or environment conditions. WSN is built of nodes which are used to collect data such as temperature, pressure, motion and pollutants, passing through the network to the main server. The development of WSN was motivated by military demanding, and today such networks are used in many industrial and consumer applications, such as fresh food tracking, Patient health monitoring and industrial process monitoring.

iPack Center, Royal Institute of Technology (KTH), Sweden contributes to WSN system developing and innovation. The iPack Center is focus on designing a wireless tracking and communication platform.[1] In case to save the cost and power consumption the WAN-SAN Coherent Architecture with a dual-layer wireless topology is used. The topology includes three main parts, wireless tracker master sensor node, low cost sensing slave nodes and Operation Center (database and server). Under the WAN layer, all the master nodes (MSN) can communicate with the OC through Wireless Wide Area Network (WAN). In the SAN (Sensor area network) layer, slave nodes are managed by one master node. Slave node collect data and all these data is gathered by that main node though RFID wireless communication.[2] Figure 1.1 shows the topology of the WSN.

1.1 Motivation

Under this existing WSN system, user collects data only from Operation Center Server. However this kind of traditional collection method has some disadvantages. It's not so easy to use, because user need some basic network knowledge to download data from server. Every sub sensor node is dependent on main node and server (database), user cannot get data directly from sensor node. Also it is not easy to change sensor data collection parameters inside the slave node by user.So iPack Center wants to enhance and improve the data fetching and parameters changing. Design a more simple, safely, easily and intuitively way to get data from sensor nodes or change nodes parameters inside is demanding. Near Field Communication (NFC)

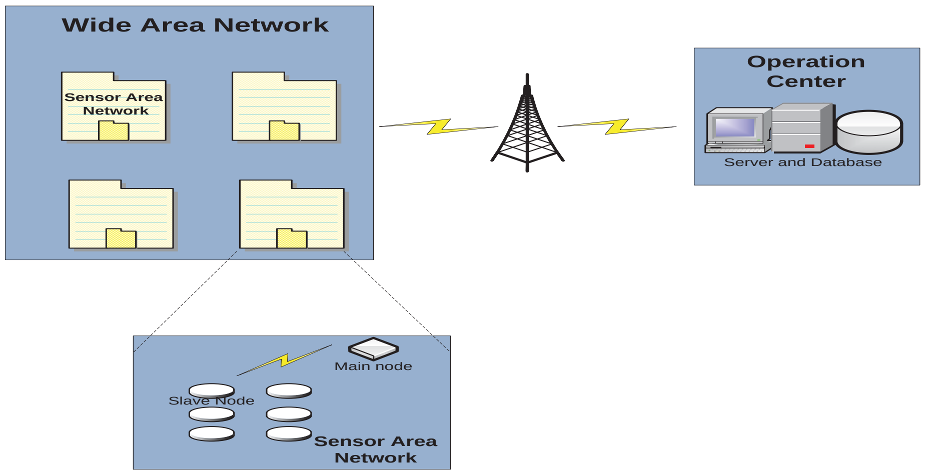


Figure 1.1. Topology of iPack Center WSN

is a short-range wireless connectivity technology that provides intuitive, simple, and safe communication between electronic devices.Communication occurs when two NFC-compatible devices are brought within four centimeters of one another. [3] This technology fulfil all the requirements. So the master thesis project is to design a NFC enable system in WSN. With this system, a extremely user friendly system is established, user can use this NFC system to fetch data directly from sensor node by a tablet or smart phone. [4]

1.2 Outline of the Thesis Report

There has total four parts in this thesis report. The first part is motivation of this thesis topic. The second part is background study, which includes basic knowledge, and project environment. For the basic knowledge chapter, we do literature study for having a basic idea of NFC protocol standards, working modes. Also we compare different wireless transmission method. In project environment chapter, both hardware and software used in the thesis are detailed introduced. The third part is the most important of the report. It detailed describe how to implement, test and optimize the NFC-enable WSN system. The last part is a conclusion part. The results of the thesis is given and future work is also handle out.

Part I

Background Study

Chapter 2

Basic Knowledge

2.1 NFC Technology Roadmap

NFC is an open platform technology, which operates at 13.56 MHz (extends on RFID), and typically requiring a distance of 10 cm or less. It has been involved in our daily life unconsciously. NFC payment, ticketing, bluetooth pairing, everything above is benefit by NFC development.[5] Therefore several organisations create their own standards. These standards determine not only the "Contactless" operating environment, such as the physical requirements of the antennas, but also the format of the data to be transferred and the data rates for that transfer. [5] Among all these standards, this thesis report is focus on the standards defined by the International Organization for Standardization (ISO), Ecma International and Sony.

2.1.1 ISO 14443-A/B, FeliCa

ISO 14443 is a standard defines polling for PICCs (Proximity Card) entering the field of a PCD(Proximity Coupling Device).[6] So ISO 14443 is just a standard for short-range contactless cards. Actually not only NFC, but also other wireless communication cards uses this standard, such as Rf protocol. ISO 14443 has some minor standardisation, such as 14443-A, 14443-B. The most wildly used traffic cards, attendance cards are mifare smart card which is ISO 14443 A standard PICC. FeliCa standard is created by Sony, and it is not a international standard product. It also works on 13.56 MHz and 10 cm operation distance. The main difference between ISO 14443 and FeliCa are their Read/write speed. ISO 14443 transmission speed is 106Kbit/s, and FeliCa speed is 212Kbit/s or 424Kbit/s.[7]

2.1.2 ISO/IEC 18092 or ECMA-340

ISO/IEC 18092 is the first formal standard for NFC, which defines communication modes for Near Field Communication Interface and Protocol (NFCIP-1) using inductive coupled devices operating at the centre frequency of 13.56MHz for devices. This standard is create for device communicate with device in peer-to-peer mode.

Both active and passive communication modes of NFCIP-1 are defined for realization a communication network. This standard specifies RF interface, initialization, transporting protocol and data exchanging methods. Ecma-340 standards is harmonised with ISO/IEC 18092 and refers to the NFCIP-1 test method standards ECMA-356 and ECMA-352. So ISO/IEC 180092 and ECMA-340 are almost the same standards. [8]

When under passive communication mode, NFCIP-1 compliant to ISO14443-A at 106Kbit/s and FeliCa at 212,424Kbit/s data rate during the initialization and selection. So NFCIP-1 compliant devices can act as a reader/writer or emulate a contactless smart card compliant with ISO 14443-A or FeliCa protocols.

When you design a NFC device or application, ISO/IEC 18092 is the most important standards. So the thesis project is completely refer to this standard when design the NFC-enable WSN system and choose the NFC chips.

2.1.3 Working Mode

NFCIP-1 has three working modes to fulfil different user requirements, and Data transmission speed ranges from 106Kbit/s to 424Kbit/s.

Peer-to-Peer mode:in this mode, two NFC devices can communicate together, one device acts as an initiator ,another device is the target. To start the communication with the target either in active or passive communication mode, an initiator shall continuously for the presence of an external RF field. Under active communication mode, both devices generate their own RF field. So in this mode, both devices need have power supply. While under passive mode, only initiator device generates RF field, the target device gets its operating power from initiator's RF field, which makes target device doesn't need power supply. [9]

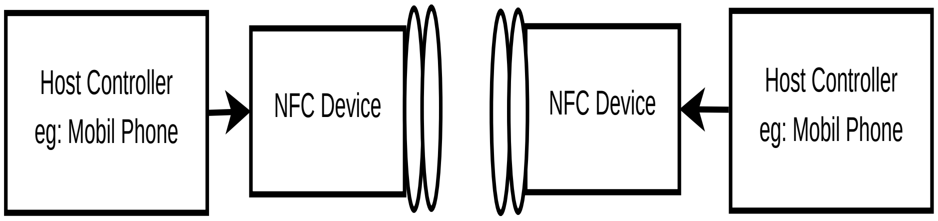


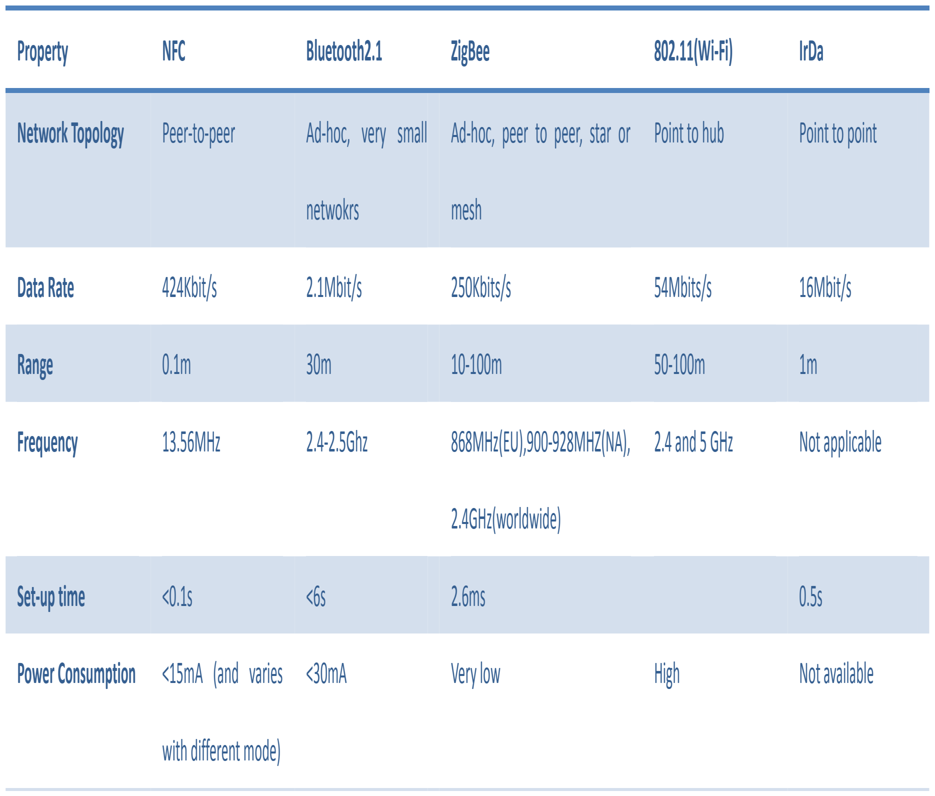
Figure 2.1. p2p communication figure

Read/Write mode:in this mode, NFC device can read/write data from/to a contactless smart card which compliant with ISO14443 or FeliCa protocols.

NFC Card Emulation mode:in this mode, one device sees another NFC device as a contactless card.

2.2 The Comparative on wireless transmission

Much different wireless technology has been developed to transmit data. And only by comparing them, We can decided which technology will be the most suitable in our wireless sensor network. Let's take a look at Figure 2.3. We list some of most popular wireless technology, NFC, Bluetooth2.1, ZigBee, WiFi and Irda. [10][11]



Property	NFC	Bluetooth2.1	ZigBee	802.11(Wi-Fi)	IrDa
Network Topology	Peer-to-peer	Ad-hoc, very small netwokrs	Ad-hoc, peer to peer, star or mesh	Point to hub	Point to point
Data Rate	424Kbit/s	2.1Mbit/s	250Kbits/s	54Mbits/s	16Mbit/s
Range	0.1m	30m	10-100m	50-100m	1m
Frequency	13.56MHz	2.4-2.5Ghz	868MHz(EU),900-928MHZ(NA), 2.4GHz(worldwide)	2.4 and 5 GHz	Not applicable
Set-up time	<0.1s	<6s	2.6ms		0.5s
Power Consumption	<15mA (and varies with different mode)	<30mA	Very low	High	Not available

Figure 2.2. Wireless Comparative

2.2.1 Bluetooth2.1

Both NFC and Bluetooth are short range communication technologies. As focus on more detail technical on table, NFC has shorter working distance than Bluetooth, which reduces the unwanted interception. With NFC, instead of performing manual configuration to identify devices. Connection between two NFC devices are automatically and very fast. So when there has more than one sub note in the WSN, NFC is more suitable than Bluetooth. Also NFC has lower power consumption than Bluetooth, which is another important features for sensor node. Less power consumption means more sensor life time. Though Bluetooth has faster data transfer speed, NFC has much more advantages.

2.2.2 ZigBee

ZigBee wireless technology is a standard enabling control and monitoring capabilities for industrial and residential applications within a +100 meters range. In contrast to ZigBee, NFC has faster transmitting speed, and almost same set-up time and power consumption. So ZigBee can be displaced by NFC in this system design

2.2.3 802.11(WiFi)

Though WiFi has much more faster transmitting time, it cost more power at the same time. WiFi is a point to hub network topology, as we want to design a peer to peer connecting system, so WiFi is not best choice.

2.2.4 IrDa

IrDa is a short range ($<$ 1 meter), line-of-sight communication standard for exchange of data over infrared light. IrDa is a point to point network topology, and has fast transmitting speed. But IrDa need a direct line of sight to connect two device, and a little long (0.5s) set-up time. So IrDa is not the best choice of design this system.

Chapter 3

Project Environment

3.1 Hardware Environment

3.1.1 Tablet with NFC Chip

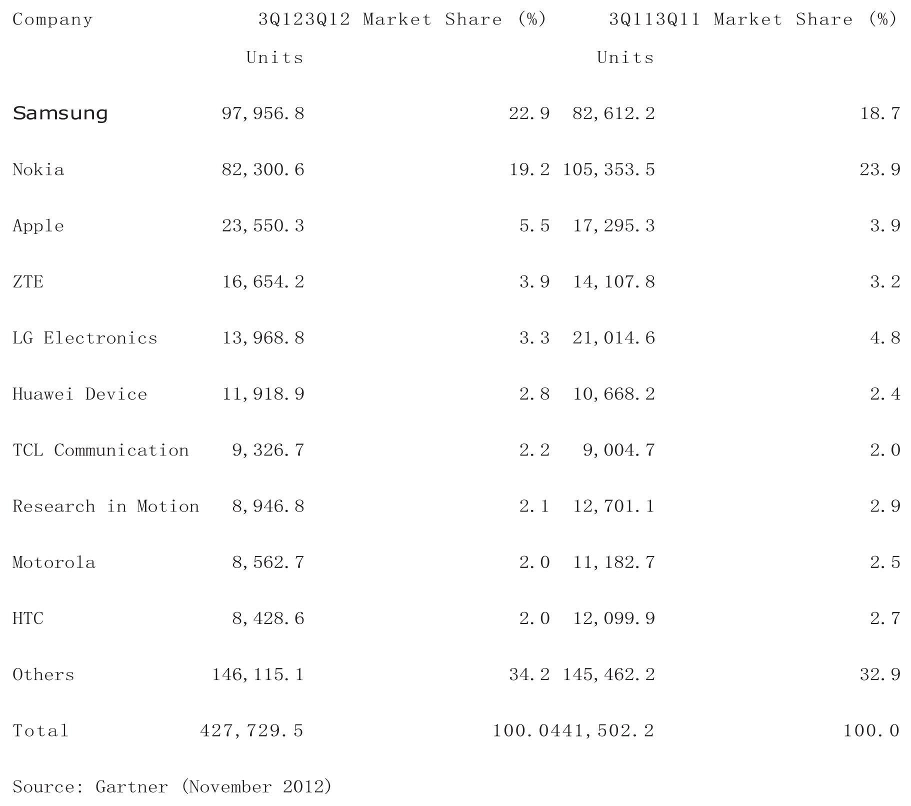
Now there has more and more cell phones and tablets shipped with NFC chip, Black-Berry, Google(Galaxy note, Nexus), HTC, Motorola, LG, Nokia(Lumia), Samsung and others company. It becomes so popular to integrate NFC into mobile device. With the growing number of NFC equipped phones, more people can be access to this technology and enjoy the convenient NFC brings to them. Such as Google Wallet, MasterCard Paypass, blue tooth pairing, sharing business cards and other mobile NFC payment method. And we think NFC enable mobile/tablet can performance better. We try to integrate a NFC equipped tablet into existing WSN system to simplify and speed up the process of controlling and monitoring, and without changing any of the WSN system. That means the NFC enable part is a additional part, and user can choose to use it or not. They don't need to give up the old WSN system to build a new system to suit this NFC enable system. So it will be more acceptable.

3.1.2 NFC Adapter

During designing the NFC-enable WSN system, the most important part is the NFC adapter. Using a pair of NFC adapter, we can make a old ordinary WSN system to be a new NFC enable WSN system. The NFC adapter is consist of one NFC chip, one programmable chip and some general connection port. The NFC chip is the core of the adapter which support software stack for NFC communication in WSN systems. while the programmable chip controls the NFC chip initial and working mode. With various connection port the NFC adapter should be easy to connect either sensor node or NFC-enable mobile/tablet.[12]

NFC Chip Choosing

In this project, the most important part of the NFC adapter is the NFC chip. To ensure the NFC chip can communicate with the most of the tablets or mobiles in the future, we need consider not only the NFC chip supporting protocol but also the software stack it offering. Look at figure 3.1. As the worldwide sales of mobile phones to end users research by Gartner (the world's leading information technology research and advisory company), we can see Samsung and Nokia were the best mobile sellers in 2012. [13] They shared almost half of the market. And referring to Chipworks and NFCtimes reported that NXP sells NFC chips to Samsung, Nokia, Google and other cellphone makers. The smartphone maker that NXP doesn't supply with NFC chips is Apple, which has not adopted NFC until now. [14] [15] After these research, we find the most of NFC enable cellphones/tablets using NXP NFC chips. So we decided choose NFC chips made by NXP semiconductors to make the NFC adapter.



Company	3Q12 Units	3Q12 Market Share (%)	3Q11 Units	3Q11 Market Share (%)
Samsung	97,956.8	22.9	82,612.2	18.7
Nokia	82,300.6	19.2	105,353.5	23.9
Apple	23,550.3	5.5	17,295.3	3.9
ZTE	16,654.2	3.9	14,107.8	3.2
LG Electronics	13,968.8	3.3	21,014.6	4.8
Huawei Device	11,918.9	2.8	10,668.2	2.4
TCL Communication	9,326.7	2.2	9,004.7	2.0
Research in Motion	8,946.8	2.1	12,701.1	2.9
Motorola	8,562.7	2.0	11,182.7	2.5
HTC	8,428.6	2.0	12,099.9	2.7
Others	146,115.1	34.2	145,462.2	32.9
Total	427,729.5	100.0	441,502.2	100.0

Source: Gartner (November 2012)

Figure 3.1. Worldwide mobile sales 2012

NFC module

NXP develops several NFC reference boards to help developers make the implementation of NFC. So in this project, we choose to buy some reference NFC boards instead of making our own PCB. In order to make the NFC adapter, the reference board should have a NFC chip, an antenna and a connection interface.

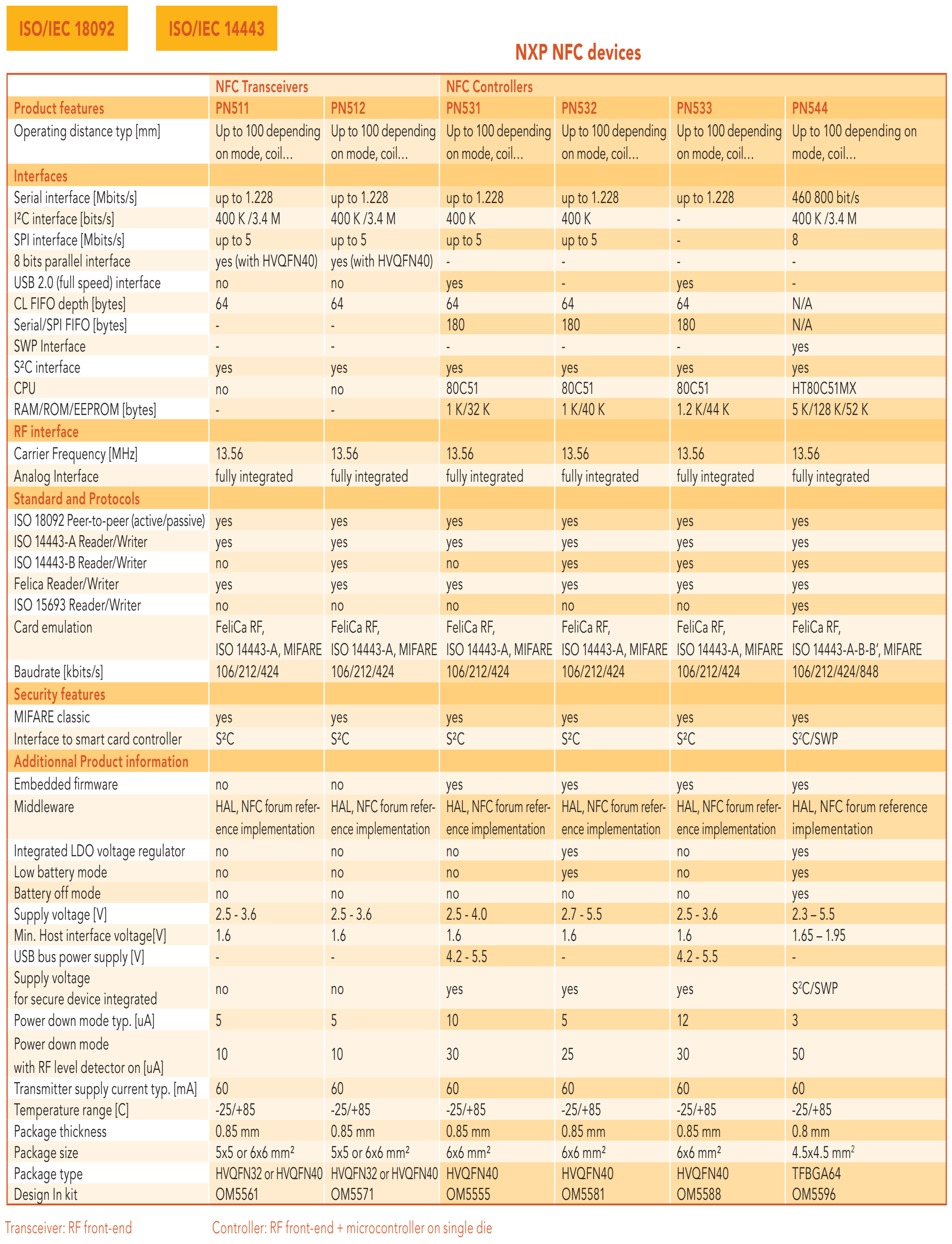
NXP chip comparison

A comparison of different NXP Semiconductor NFC chips can be seen in detail in figure 3.2. We can easily find there has two different NFC chip modules, NFC transceivers and controllers. NFC Controllers is a highly integrated transmission module including a microcontroller. The memory (RAM, ROM, EEPROM) are only available on controllers module. The 80C51 MCU used in controller adds the embedded firmware. This easy-to-use firmware combines a modulation and demodulation concept for different supported modes and required host controller interfaces at 13.56 MHz. Also in many applications a high level language is needed, this requires a MCU can transmit the commands to the NFC module. The first part of the figure is interfaces. The embedded firmware and the internal hardware support different interfaces (I^2C, SPI, USB 2.0 and serial), which can be handled by host controller protocol. In this project we prefer NFC controllers, because of its easy-to-use firmware. To ensure NFC controller can connect to any sensor node, we need controller support as much interfaces as possible. So we find only PN531, PN532 and PN544 can meet these requirements.

The RF and security part is useful for some specific applications. And all the NFC chips offered by NXP, have the same RF interface and security features. However they have very small details difference of The standard and protocols. Among all the controllers, PN532, PN533 and PN544 support more mainstream standard than PN531. Supporting more means it has more chance to meet tablet/mobile NFC chips' protocol, and ensure our NFC chip can communicate with all kinds of NFC-enable tablet/mobile later. For this thesis project, we need our NFC chip supports ISO18092/ECMA-340 standards to do peer-to-peer communication. The good news is, all these NFC chips support ISO 18092 peer-to-peer (active/passive) protocols.

The last part is power consumption of NFC module and secure part. This part is very important, because in this project, the module should be connect to sensor node. And power is always a sensitive issue to sensor nodes. Less power consumption means longer life time of the wireless sensor network. PN544 and PN532 consume less power during power down mode and support low battery mode.

After comparing all the NXP NFC chips, PN532 and PN544 are our project's best choice. So we want to purchase the reference board with NFC chip PN532 or PN544. As the PN544 has faster I2C, SPI communication speed and lower power down mode power consumption than the PN532, we choose PN544 first. But after search on internet, we found there has no stock recently. At last we choose PN532



ISO/IEC 18092 ISO/IEC 14443

NXP NFC devices

	NFC Transceivers		NFC Controllers			
Product features	PN511	PN512	PN531	PN532	PN533	PN544
Operating distance typ [mm]	Up to 100 depending on mode, coil...	Up to 100 depending on mode, coil...	Up to 100 depending on mode, coil...	Up to 100 depending on mode, coil...	Up to 100 depending on mode, coil...	Up to 100 depending on mode, coil...
Interfaces						
Serial interface [Mbits/s]	up to 1.228	up to 1.228	up to 1.228	up to 1.228	up to 1.228	460 800 bit/s
I²C interface [bits/s]	400 K /3.4 M	400 K /3.4 M	400 K	400 K	-	400 K /3.4 M
SPI interface [Mbits/s]	up to 5	up to 5	up to 5	up to 5	-	8
8 bits parallel interface	yes (with HVQFN40)	yes (with HVQFN40)	-	-	-	-
USB 2.0 (full speed) interface	no	no	yes	-	yes	-
CL FIFO depth [bytes]	64	64	64	64	64	N/A
Serial/SPI FIFO [bytes]	-	-	180	180	180	N/A
SWP Interface	-	-	-	-	-	yes
S²C interface	yes	yes	yes	yes	yes	yes
CPU	no	no	80C51	80C51	80C51	HT80C51MX
RAM/ROM/EEPROM [bytes]	-	-	1 K/32 K	1 K/40 K	1.2 K/44 K	5 K/128 K/52 K
RF interface						
Carrier Frequency [MHz]	13.56	13.56	13.56	13.56	13.56	13.56
Analog Interface	fully integrated	fully integrated	fully integrated	fully integrated	fully integrated	fully integrated
Standard and Protocols						
ISO 18092 Peer-to-peer (active/passive)	yes	yes	yes	yes	yes	yes
ISO 14443-A Reader/Writer	yes	yes	yes	yes	yes	yes
ISO 14443-B Reader/Writer	no	yes	no	yes	yes	yes
Felica Reader/Writer	yes	yes	yes	yes	yes	yes
ISO 15693 Reader/Writer	no	no	no	no	no	yes
Card emulation	FeliCa RF, ISO 14443-A, MIFARE	FeliCa RF, ISO 14443-A, MIFARE	FeliCa RF, ISO 14443-A, MIFARE	FeliCa RF, ISO 14443-A, MIFARE	FeliCa RF, ISO 14443-A, MIFARE	FeliCa RF, ISO 14443-A-B-B', MIFARE
Baudrate [kbits/s]	106/212/424	106/212/424	106/212/424	106/212/424	106/212/424	106/212/424/848
Security features						
MIFARE classic	yes	yes	yes	yes	yes	yes
Interface to smart card controller	S²C	S²C	S²C	S²C	S²C	S²C/SWP
Additionnal Product information						
Embedded firmware	no	no	yes	yes	yes	yes
Middleware	HAL, NFC forum reference implementation	HAL, NFC forum reference implementation	HAL, NFC forum reference implementation	HAL, NFC forum reference implementation	HAL, NFC forum reference implementation	HAL, NFC forum reference implementation
Integrated LDO voltage regulator	no	no	no	yes	no	yes
Low battery mode	no	no	no	yes	no	yes
Battery off mode	no	no	no	no	no	yes
Supply voltage [V]	2.5 - 3.6	2.5 - 3.6	2.5 - 4.0	2.7 - 5.5	2.5 - 3.6	2.3 – 5.5
Min. Host interface voltage[V]	1.6	1.6	1.6	1.6	1.6	1.65 – 1.95
USB bus power supply [V]	-	-	4.2 - 5.5	-	4.2 - 5.5	-
Supply voltage for secure device integrated	no	no	yes	yes	yes	S²C/SWP
Power down mode typ. [uA]	5	5	10	5	12	3
Power down mode with RF level detector on [uA]	10	10	30	25	30	50
Transmitter supply current typ. [mA]	60	60	60	60	60	60
Temperature range [C]	-25/+85	-25/+85	-25/+85	-25/+85	-25/+85	-25/+85
Package thickness	0.85 mm	0.85 mm	0.85 mm	0.85 mm	0.85 mm	0.8 mm
Package size	5x5 or 6x6 mm²	5x5 or 6x6 mm²	6x6 mm²	6x6 mm²	6x6 mm²	4.5x4.5 mm²
Package type	HVQFN32 or HVQFN40	HVQFN32 or HVQFN40	HVQFN40	HVQFN40	HVQFN40	TFBGA64
Design In kit	OM5561	OM5571	OM5555	OM5581	OM5588	OM5596

Transceiver: RF front-end

Controller: RF front-end + microcontroller on single die

Figure 3.2. NXP NFC chip competition

as our NFC chip to make the NFC adapter at last.

3.1.3 PN532 features and benefits

The PN532 ia a highly integrated NFC controller module for contactless communication at 13.56 MHz. It supports ISO/IEC 14443-A(mifare)/B, FeliCa and NFCIP-1 protocols. The communication distance of NFCIP-1 up to 50 mm depending on antenna size. The integrated RF interface for NFCIP-1 up to 424 kbit/s,however with external analog components higher data rate speed can be reached. The PN532 supports three different host interface, SPI, I2C and High-speed UART. For low power consumption, the PN532 supports Soft-Power-Down mode and Hard-Power-Down. There has five main parts of the PN532, the 80C51 MCU, Host Interface, Power Distribution, Power Clock Reset Controller (PCR) and Contactless Interface unit (CIU). The PN532 block can be seen on Figure 3.3.

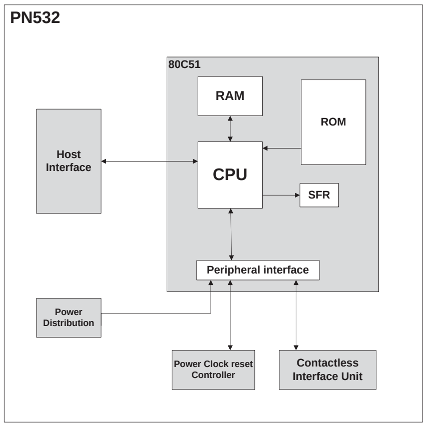


Figure 3.3. PN532 Block Description

80C51

The microcontroller 80C51 with 40 KB ROM and 1 KB RAM, embedded in the PN532 using an external 27.12MHz oscillator for time reference. When we got the PN532 and all the technical manuals from NXP Semiconductor, we found there is no debug port for 80C51 on the PN532. And the PN532 firmware is already integrated to the memory. The PN532 memory is composed of two main spaces: data memory in RAM and program memory in ROM. Data memory is divided into 2 parts: 384-byte IDATA (258-byte RAM and 128-byte SFR) with byte-wide addressing, 64 KB extended XRAM with 2-byte-wide addressing. NXP only allows the PN532 user overwrite the content of SFR and XRAM registers.[16][17] That means we could not write our code to the RAM of 80C51 to control the PN532 directly. After reading the PN532 user manual, we found the PN532 could be controlled though communication Interface (High-speed UART, SPI, I2C). Thus we decided to write the PN532 controlling code on the other MCU, and connect it with the PN532. Then we can control the PN532 indirectly.

Contactless interface Unit and Transparent Transmission

The PN532 CIU is a modem for contactless communication at 13.56 MHz. It supports six different operating modes: ISO/IEC 14443A/Mifare reader/writer, FeliCa reader/writer, ISO/IEC 14443B reader/writer, ISO/IEC 14443A/Mifare card 1K or Mifare 4K card emulation mode, Felica card emulation and ISO/IEC 18092, ECMA 340 NFCIP-1 peer-to-peer. It offers the possibility to communicate to another NFCIP-1 compliant device at up to 424 kbit/s data rate. That means the CIU can communicate with the tablet or mobile phone with NFC chips compliant to ISO18092 protocol.

There is a highly integrated analog circuitry to demodulate and decode received data to the NFCIP-1 mode communication scheme. And with the cooperation with the firmware inside memory, the PN532 can transform the UART receiving data to compliant NFCIP-1 protocol. With this feature the PN532 adopt transparent transmission protocol, when it is peer-to-peer communicating with another PN532 chip. It can save us a lot of working on transform data to NFCIP-1 protocol or NFIP-1 data to normal ascii data.

An 64*8 bits send and receive FIFO buffer is implemented in the CIU. The FIFO buffer allows a convenient data transfer from the 80C51 to the CIU and vice versa.

3.1.4 PN532 Demo Board

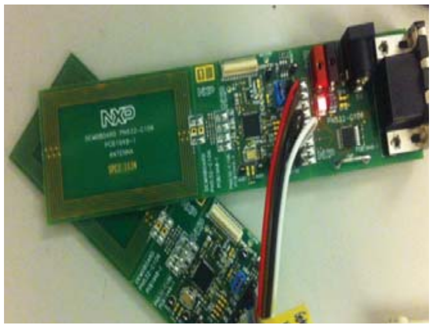


Figure 3.4. Pn532 demo board

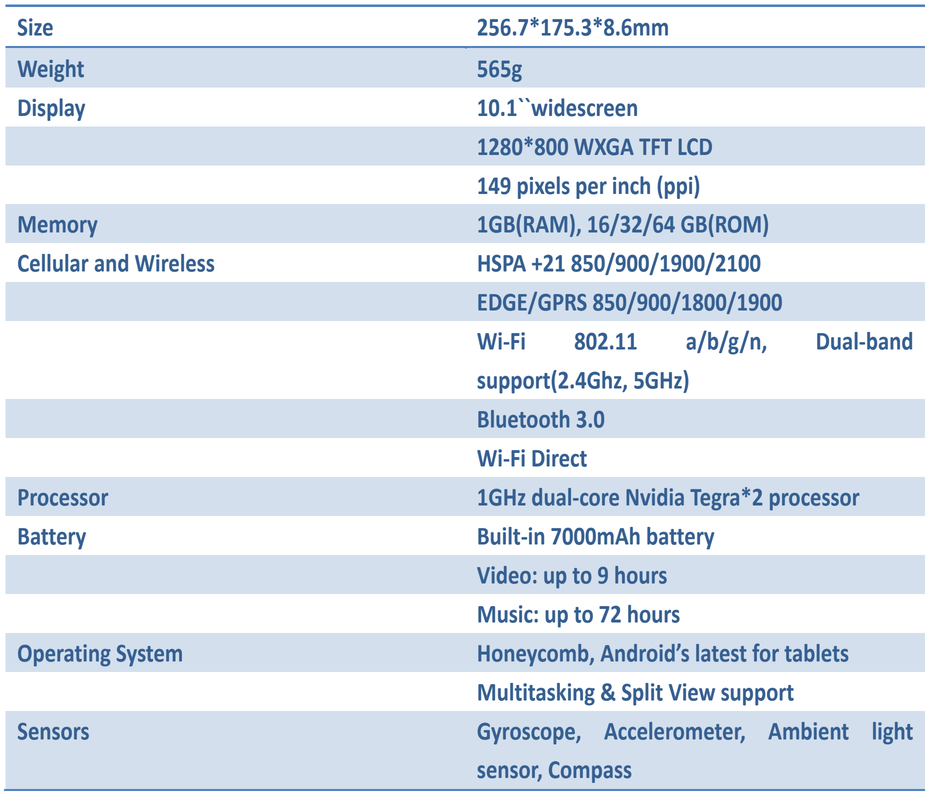
To use the PN532 chip, there had two choice for us. The first one was that we bought the PN532 chip only. Then we designed the PN532 application board by ourself which required PCB board, Antenna designing and functionality testing. That took too much time for the thesis project, and making very few PCB board was also expensive. The second way was we purchase the PN52 demo board from NXP Semiconductor to simply and speed up the thesis project.

In the end, we chose to purchase the demo board PC1948-1 for this project. Board PCB1948-1 is a demo board recommended by NXP, for PN532 NFC chip application design. The interface with the host controller of this board is a high speed UART (HSU) and allowed power supply voltage from 2.7v to 5.4v. There has no SPI or USB connecting port on this board. However it is possible to break the PCB, removing the interface and power supply part, in order to connect it to a host controller with a different interface(SPI or USB) and power sources. The demo board PCB is split into four parts. First part is power supply. There has two supply voltage are used by the PN532: V_{BAT} and P_{VDD}. V_{BAT} must be between 2.7V and 5V; P_{VDD} must be between 1.6V and 3.6V. On the demo board, V_{BAT} and P_{VDD} are connected, but it is possible to break the board to disconnect them. V_{BAT} is for battery supply, and P_{VDD} is for another supply voltage. In this project, we choose battery as power supply for NFC adapter, so V_{BAT} is used on this demo board. The second part of the demo board is the main part (containing PN532 IC). The PN532 IC is the core of this board, it supports contactless communication using several different protocols. The third part is antenna matching components part and the last is the antenna itself.[18][16]

3.1.5 Galaxy Tab

Figure 3.5. Galaxy Tab

The project uses one sensor node as a host controller and a tablet as another host controller, and both host controllers will be connected with the NFC adapters. For the task of WSN data monitoring and sensor node parameter changing, a giant display is needed. Also the platform should be easy to buy and sells well on the market. The Samsung Galaxy Tab 10.1 fulfills all the above requirements, so we choose it as the demo's tablet. It has almost the biggest screen with high resolution among all tablet products. Also it has 3G, Wireless connection and bluetooth and many build-in sensors. More detail specification of Galaxy Tab can be seen on figure 3.5 To design the demo system, another master student is focus on design a User-



Size	256.7*175.3*8.6mm
Weight	565g
Display	10.1``widescreen
	1280*800 WXGA TFT LCD
	149 pixels per inch (ppi)
Memory	1GB(RAM), 16/32/64 GB(ROM)
Cellular and Wireless	HSPA +21 850/900/1900/2100
	EDGE/GPRS 850/900/1800/1900
	Wi-Fi 802.11 a/b/g/n, Dual-band support(2.4Ghz, 5GHz)
	Bluetooth 3.0
	Wi-Fi Direct
Processor	1GHz dual-core Nvidia Tegra*2 processor
Battery	Built-in 7000mAh battery
	Video: up to 9 hours
	Music: up to 72 hours
Operating System	Honeycomb, Android's latest for tablets
	Multitasking & Split View support
Sensors	Gyroscope, Accelerometer, Ambient light sensor, Compass

Figure 3.6. Specifications of Galaxy Tab 10.1

Centric ubiquitous intelligence base on open mobile computing platform for WSN. The UCUI system is contribute to the NFC communication Application of tablet. The application lets user using tablet getting data from WSN or changing parameter of WSN more easily. The operating system of Galaxy Tab is a customized Android OS of Samsung. Android is one of the most popular OS now, it offers plenty of APIs from Android. Also Samsung offers adds-on library. These speed up and simplify the UCUI developining. The tablet has a dock, which provides a USB slave port for the data exchange with PC. And it also supports for an external USB host port with the NFC adapter.

3.1.6 slave node(MSP430)

Ipack Center is focus on developing a wireless tracking and communication platform. The NFC adapter developed in this project is used on the slave node of Ipack Center wireless sensor platform. The slave node is focus on low power and energy scavenging issues, low cost in hardware implementation.[1] The newest slave node SSN-v3.8 is using MSP430F2132 to collect data from sensors, process the data, and transfer the data to main node. The Texas Instruments MSP430F2132 is an ultra-low-power microcontroller with two build-in 16-bit timers, a fast 10-bit A/D converter with integrated reference and a data transfer controller, a comparator, built-in communication capability using the universal serial communication interface, and up to 24 I/O pins.[19] But the SSN-v3.8 PCB hasn't lead out either UART or SPI interface on the PCB. Because the NFC adapter needs UART or SPI to connect the sensor node, the SSN-v3.8 PCB cannot be used as demo slave node in this project.

3.1.7 FTDI VNC

Figure 3.7. V2DIP1-32

VNC2 is a chip of Future Technology Devices International LTD(FTDI). It has nine peripheral interface modules: Debugger Interface, UART, PWM, FIFO, SPI master, SPI slave, GPIO and general purpose timers. It is designed to be bridge between two or more interfaces. VNC2 is mainly used as a USB bus controller, because of supporting two USB ports which can be set as host or slave mode individually. Furthermore, it has one UART peripheral interface (baud rate from 300 baud rates to 6M baud rates), two SPI slave and one SPI master interface. With the chip, data can be transmitted from one bus to another very easily. The VNC2 block diagram can be seen on Diagram 14. Besides VNC2 is a programmable SoC device with a powerful embedded microprocessor. It has a 16Kbytes on chip RAM and 256Kbytes(128K*16-bit) on chip E-Flash memory. The processor is based on proprietary 16-bit Harvard architecture with separate code and data space. VNC2 use a 12MHz external oscillator. And with an internal PLL, three operating frequencies are available(12MHz, 24MHz and normal operation of 48MHz). [20][21]

The VNC2 RTOS(VOS)is a pre-emptive priority-based multi-tasking operating system. VOS is developed by FTDI and provides to customers for free of charge. VOS supports up to 31 customized priorities (from 1 to 31), and 1 idle task (number 0). Higher priority with a higher value. VOS supports more threads, and several threads can share the same priority. They run in round robin fashion. VOS also supports for mutex, software timer, semaphore, and critical section execution. It is a small but powerful RTOS.

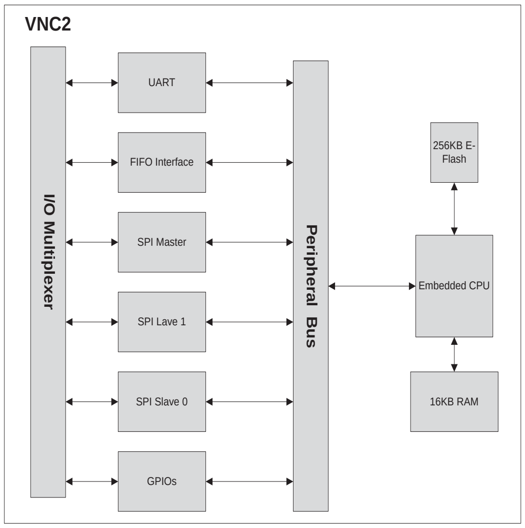


Figure 3.8. VCN2 block Diagram

VNC2 is a low cost microcontroller supports power saving modes and standby mode. To save power, it can reduce the operating frequencies down to 12MHZ. When a particular peripheral is not used, it is powered down internally thus saving power. The firmware can control the VNC2 into standby mode, with no clocks running or system blocks powered. The device will wake up by toggling any of the USB, SPI or UART ring indicator signals. So due to its powerless feature, VNC2 also can be used as sensor node.

V2DIP1-32 module is demo board designed using VNC2-32Q IC. V2DIP1-32 module allow rapid development of design using the VCN2 IC. It provides a basic firmware for users, and users also can also develop own firmware. The module also provides access to the UART, SPI and GPIO interface pins of the VNC2 IC via its IO bus pins. The module can be seen on diagram 13. [22]

3.2 Software Environment

3.2.1 PC RS232 COM port programme IDE

Visual Studio 2010 is an IDE from Microsoft. It is used to develop console and graphical interface applications for all platforms supported by Microsoft, Windows Mobile and Windows CE. All the coding, compiling and debugging tasks can be done in this software. Visual Studio supports different programming languages by means of language services, which allows the code editor and debugger to support nearly any programming language. It provide build-in languages (such as C/C++, VB.NET) and user also can install languages (such as M, Python, Ruby) separately.

In this project, because of the strategy we using to implement the NFC adaptor code, we needed a very stable platform to write and test our code at first. Our own build VNC2 firmware was tested, and it worked good. But testing times were not enough, we still doubted the stability of the firmware would influence the testing result of our code. So we chose Visual Studio as the IDE, Windows X86 as the platform to write and debugging our code. We wrote the code, debugged and released it as a console application. The application used PC RS232 port to send/receive command from/to the PN532 demo board. Only when the application could control two PN532 demo board doing the NFCIP-1 peer-to-peer communications with amount of randomly data, we could ensure our code was right. Then we integrated the code to VNC2 platform to implement the NFC adapter demo.

3.2.2 Serial port Monitor and Sniffer Software

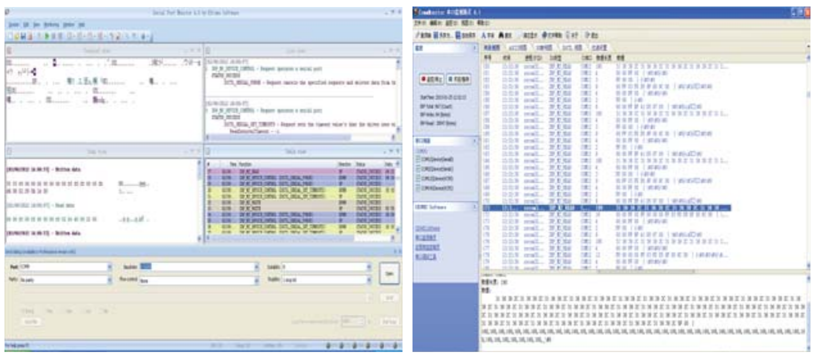


Figure 3.9. Serial Port Monitor, CommMonitor

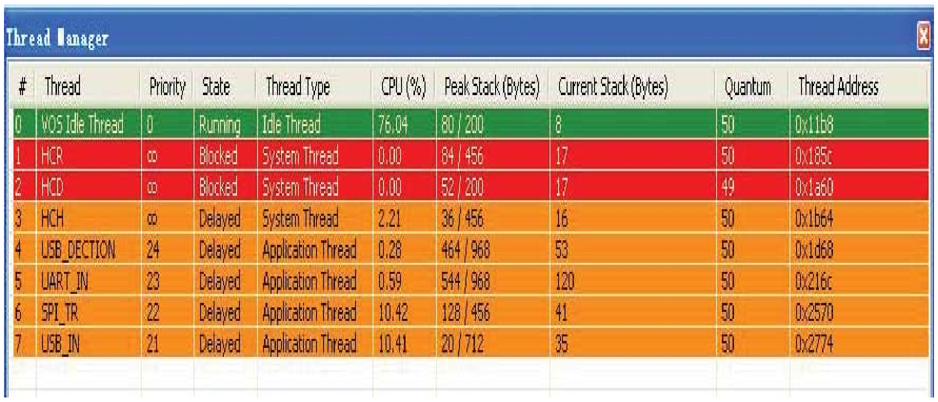
There had two serial port monitor software used during the thesis work. They are Serial Port Monitor and CommMonitor. Neither of these two software is perfect, they have their own advantages.

Serial Port Monitor (SPM) is a COM port monitoring software of Eltima software. And CommMonitor is a also COM port monitoring software of CEIWEI Software technology company. Both of these two software are powerful system utility for RS232 COM ports monitoring. They can connect to a COM port, which is even already opened by any application, and can start sniffing simultaneously. They will show up any data go through the COM port. Everything is captured in real-time. Moreover data can be viewed in all four different ways at the same time: table, line, dump and terminal mode. Each providing a different way to represent recorded serial data. The data sniff by the them also can in various formates (string, octal, decimal, hexadecimal, mixed). Thus we can monitor any special commands and data with the software.

CommMonitor is used more than Serial Port Monitor, because it can sniff several COM ports simultaneously. And it supports ASCII display better. Some times Serial Port Monitor cannot transform HEX to ASCII. However SPM has better results storing mechanism and GUI. SPM can store sniffer results to .spm and can open .spm by itself next time. CommMonitor can only save results to text file, when you want to review the results it is always indistinct. So we need to use both of them depends on the situation at the moment.

3.2.3 VNC2 IDE

The program for VNC2 is written in C programming language. Basic C grammar is fully supported, besides FTDI also provides some C libraries for VNC2, such as stdio, stdlib, and sting. Also full hardware drivers are supported. All these libraries are integrated in IDE called Vinculum II IDE together with a C compiler, an asm assembler, a linker, and a debugger. VNC2 IDE debug tools are very useful for



#	Thread	Priority	State	Thread Type	CPU (%)	Peak Stack (Bytes)	Current Stack (Bytes)	Quantum	Thread Address
0	VOS Idle Thread	0	Running	Idle Thread	76.04	80 / 200	8	50	0x11b8
1	HCR	∞	Blocked	System Thread	0.00	84 / 456	17	50	0x185c
2	HCD	∞	Blocked	System Thread	0.00	52 / 200	17	49	0x1a60
3	HCH	∞	Delayed	System Thread	2.21	36 / 456	16	50	0x1b64
4	USB_DECTION	24	Delayed	Application Thread	0.28	464 / 968	53	50	0x1d68
5	UART_IN	23	Delayed	Application Thread	0.59	544 / 968	120	50	0x216c
6	SPI_TR	22	Delayed	Application Thread	10.42	128 / 456	41	50	0x2570
7	USB_IN	21	Delayed	Application Thread	10.41	20 / 712	35	50	0x2774

Figure 3.10. Thread Manger

making a RTOS, especially the thread manager tool. The Thread manager tool Diagram can be seen on Figure 3.10. From the thread manager, we can see every thread in the RTOS and their priority, state. And when the debugging is running, we can see more information about CPU usage ratio, peak stack, current stack. The debug tools helped us a lot, when we needed to diagnose the problem of the VNC2 code.

Part II

NFC Adapter Implementation

Chapter 4

Four Ways Implementation Trade Off

To implement the function of controlling PN532 peer-to-peer communication, we need write code for VNC2 to drive the NFC adapter working. For the application, we think not only NFC hardware connection, but also NFC software integration. There has a open source project solution libnfc. And NXP Semiconductors is providing two solutions with all the needed documents and codes for their NFC chips users. Besides we also bring out one our own solutions for the implementation.

4.1 Libnfc

There has a open source library for NFC which named libnfc.[23] It provides complete transparency and royalty-free use of low level NFC SDK and programmers API for everyone. The library can be used for various RFID and NFC applications. It currently supports for ISO/IEC 14443-A/B, FeliCa, Jewel tags and Data Exchange Protocol as target and initiator. Like most of the open source, all major OS are supported, GNU/LINUX, Mac OS X and Windows. And we can find the library supports the PN532 on the supporting hardware list. Though open source library has many advantages, such as fully access to the codes, flexible, a motivated developers community. It also has a lot of problems. It is not developed for commercial users, so it sometimes not as stable as the commercial library and SDK. In addition, when you have a problem cannot be solved by yourself, you cannot get support from authentic supporting engineers. Only the developers community you can rely on. As we want to develop a stable application and we are kind of short of time, we didn't decide to use Libnfc.

4.2 NFC-FRI SDK integrate

The first recommendation of NXP is integrate the NFC-FRI(Forum Reference implementation) SDK(Software Design Kit) onto other platform. NFC-FRI SDK is basically a generic software layer and also the link between NFC application and NFC hardware. It is developed in pure C coding. The NFC-FIR stack provides several useful mechanisms: Automatic detecting NFC device, allowing user to register for different NFC event types(Application selection, NFC initiator or Target mode set), noticing user when a NFC event occurs. For the convenient, it provides API modules and libraries for user. But it doesn't allowed users fully access to the libraries in the SDK. So we can just use the API providing by the SDK. [24][25]

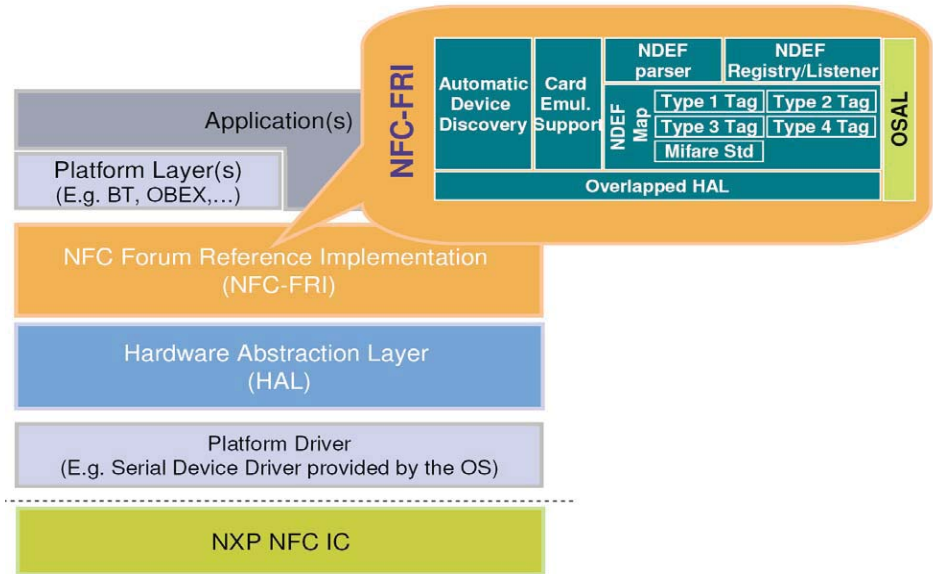
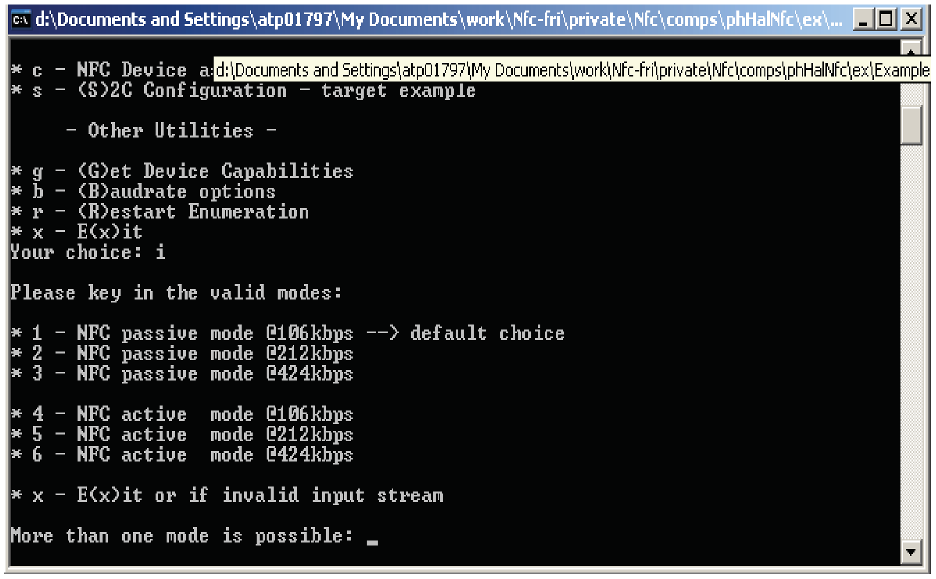

NFC-FRI
Automatic Device Discovery
Card Emul. Support
NDEF parser
NDEF Registry/Listener
NDEF Map
Type 1 Tag
Type 2 Tag
Type 3 Tag
Type 4 Tag
Mifare Std
Overlapped HAL
OSAL
Application(s)
Platform Layer(s) (E.g. BT, OBEX,...)
NFC Forum Reference Implementation (NFC-FRI)
Hardware Abstraction Layer (HAL)
Platform Driver (E.g. Serial Device Driver provided by the OS)
NXP NFC IC

Figure 4.1. overview of NFC-FRI stack integration in a system

The software has been architected in a way that the platform dependencies are located within two separate and dedicated sub-modules:the OSAL(Operating system Abstraction Layer) and the DAL(Driver Abstraction Layer). That means to integrate the NFC-FRI-SDK onto any platforms, the process consists in compiling all platform dependent and integration sources by linking with LibNFC and HalDL libraries. Platform dependent may consist of the operation system, the programming code language. So we can only use FRI SDK to develop our application for specific sensor node platform, once we changed the platform, it takes the same effort to develop another application for the new platform. Due to i-Pack Center is trying different new platforms for their sensor nodes, this solution is not a good idea.[26]

4.3 Porting NFC HAL Software

NXP Semiconductors also provides the Linux/Windows X86 HAL(Hardware Abstraction Layer) SDK stack for their PN5XX NFC chips users. It is lower level than the FRI SDK. The software also is written in pure C code. So we can use the HAL SDK to integrate the software into an application or library, or port it to another platform or operating system. To run and test the application, we install the HAL SDK v2.2 in Windows XP OS, using MS visual studio compiling and releasing the application. The application is more powerful than FRI SDK. Especially various parameters can be chosen and modified before peer-top-peer communication examples. The function include getting device capabilities, choosing different NFC working modes, setting up peer-to-peer flow control parameters and baud rate. When peer-to-peer working mode is chosen, user need to choose working as initiator or target at different data speed rate during peer-to-peer mode. The flow control setting is very useful which includes amounts of setting: payload data bytes, iterations, number of bytes per iteration subtract number of bytes per iteration and waiting time to initiator request. But the software works not very stable when we are doing the software testing. And it also doesn't allowed user to fully access to all the library files. [27]


d:\Documents and Settings\atp01797\My Documents\work\Nfc-fri\private\Nfc\comps\phHalNfc\ex\...

```
* c - NFC Device a d:\Documents and Settings\atp01797\My Documents\work\Nfc-fri\private\Nfc\comps\phHalNfc\ex\Example
* s - (S)2C Configuration - target example

     - Other Utilities -

* g - (G)et Device Capabilities
* b - (B)audrate options
* r - (R)estart Enumeration
* x - E(x)it
Your choice: i

Please key in the valid modes:

* 1 - NFC passive mode @106kbps --> default choice
* 2 - NFC passive mode @212kbps
* 3 - NFC passive mode @424kbps

* 4 - NFC active  mode @106kbps
* 5 - NFC active  mode @212kbps
* 6 - NFC active  mode @424kbps

* x - E(x)it or if invalid input stream

More than one mode is possible: _
```

Figure 4.2. overview of HAL Software

To port the NFC HAL SDK to the other OS, specific files(hardware dependent files) need to be modified. They are *phDalNfc.c* which contains the device driver adaptation functions, phcsBflBal_Hw1SerWin.c which implements the low-level I/O module of the BLF layer. Furthermore HAL SDK now only supports PN531, PN532 and PN512 NFC chips. considering to change the NFC chip to PN544 (The design board is out of order now, but it is better than PN532), we do not choose this solution to develop our application for NFC adapter.[28]

4.4 Sniffer and Write own SDK

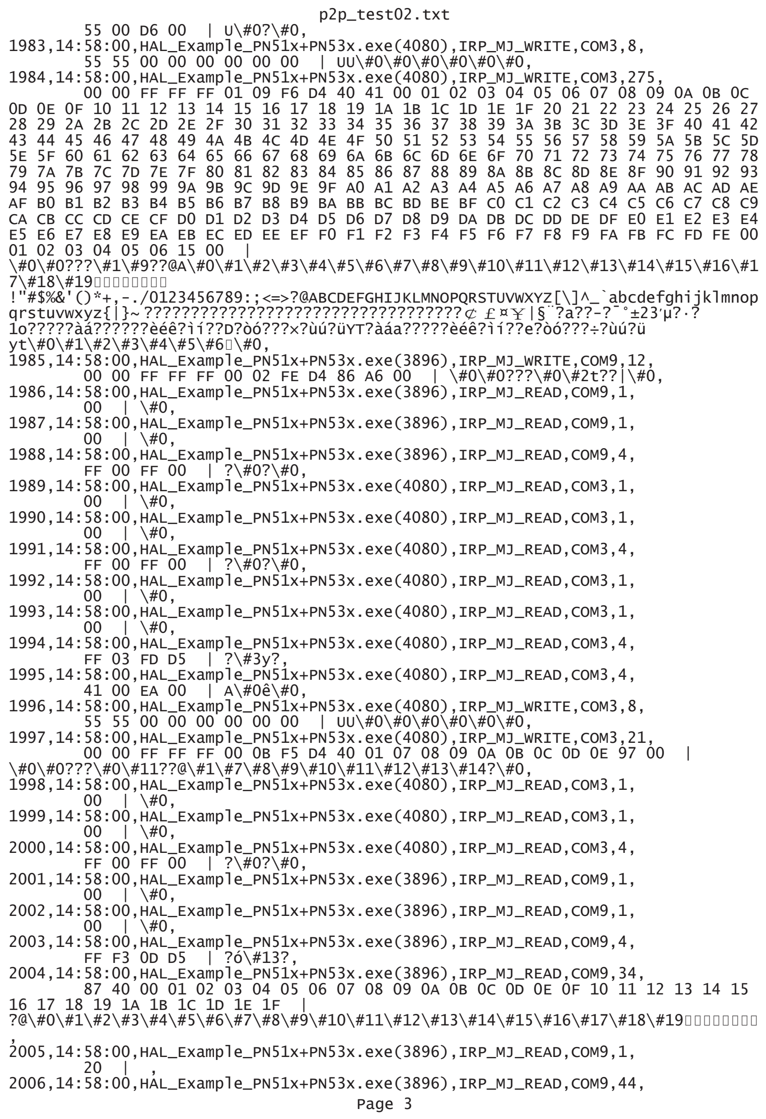
The thesis project wants a easy to use, modify and integrate SDK to design the NFC application for the PN532. The open source and the solutions recommended by NXP Semiconductor don't match the thesis project's request. So we figured out a new solution on the basis of HAL SDK.

From the technical manual studying (This thesis 3.1.3 mentioned), We already knew that with the cooperation of the firmware and the CIU, two PN532 peer-to-peer communication compliant to transparent transmission protocol. Besides we knew that the PN532 can be controlled by sending command through its communication interface. And the only communication interface on the demo board is the High-Speed UART port. Under these factors, we can write our own code for NFC application by learning from PN532 user manual, HAL software code and analysing sniffer results of HAL SDK application. The PN532 user manual describes all the details about the command frame PN532 supports. But it was still far from solid understanding the full sequence of command to control the PN532.

The most tough work in this thesis was sniffer and analyse the results. By using two UART sniff software, we could monitor all the HEX data coming/going though the PN532 at the same time when the PN532 was running the HAL SDK application. As peer-to-peer communication compliant to NFCIP-1 protocol, one PN532 works as initiator and another works as target. The flow control of initiator and target are quite different, So we sniffed both two PN532 simultaneously. For detailed understanding the sequence of different flow control setting and active/passive working mode, we sniffed and stored all these different setting results for further analysing. software.[29][30]

From figure 4.3 we can see one page sniffer command data when PN532 was running the HAL SDK application. The sniffer results includes five elements, time, application name, COM port, read-write operation, both HEX and ASCII format data. We can tell apart initiator and target receiving/sending command by the port number and read-write operation. The HEX and ASCII formate data is the content of the commands, and we need to refer the PN532 user manual to recognize them.

After analysing the results we confirmed that two PN532 communication compliant to transparent transmission protocol which is very important for later coding work. Eventually after all these preparing we can write our own SDK without official libraries.



```
                               p2p_test02.txt
        55 00 D6 00  | U\#0?\#0,
1983,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_WRITE,COM3,8,
        55 55 00 00 00 00 00 00  | UU\#0\#0\#0\#0\#0\#0,
1984,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_WRITE,COM3,275,
        00 00 FF FF FF 01 09 F6 D4 40 41 00 01 02 03 04 05 06 07 08 09 0A 0B 0C
0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27
28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42
43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D
5E 5F 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78
79 7A 7B 7C 7D 7E 7F 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93
94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE
AF B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF C0 C1 C2 C3 C4 C5 C6 C7 C8 C9
CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4
E5 E6 E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE 00
01 02 03 04 05 06 15 00  |
\#0\#0???\#1\#9??@A\#0\#1\#2\#3\#4\#5\#6\#7\#8\#9\#10\#11\#12\#13\#14\#15\#16\#1
7\#18\#19□□□□□□□□□
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnop
qrstuvwxyz{|}~??????????????????????????????????¢£¤¥|§¨?a??-?¯°±23′µ?·?
1o?????àá??????èéê?ìí??D?òó???×?ùú?üYT?àáa?????èéê?ìí??e?òó???÷?ùú?ü
yt\#0\#1\#2\#3\#4\#5\#6□\#0,
1985,14:58:00,HAL_Example_PN51x+PN53x.exe(3896),IRP_MJ_WRITE,COM9,12,
        00 00 FF FF FF 00 02 FE D4 86 A6 00  | \#0\#0???\#0\#2t??|\#0,
1986,14:58:00,HAL_Example_PN51x+PN53x.exe(3896),IRP_MJ_READ,COM9,1,
        00  | \#0,
1987,14:58:00,HAL_Example_PN51x+PN53x.exe(3896),IRP_MJ_READ,COM9,1,
        00  | \#0,
1988,14:58:00,HAL_Example_PN51x+PN53x.exe(3896),IRP_MJ_READ,COM9,4,
        FF 00 FF 00  | ?\#0?\#0,
1989,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_READ,COM3,1,
        00  | \#0,
1990,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_READ,COM3,1,
        00  | \#0,
1991,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_READ,COM3,4,
        FF 00 FF 00  | ?\#0?\#0,
1992,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_READ,COM3,1,
        00  | \#0,
1993,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_READ,COM3,1,
        00  | \#0,
1994,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_READ,COM3,4,
        FF 03 FD D5  | ?\#3y?,
1995,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_READ,COM3,4,
        41 00 EA 00  | A\#0ê\#0,
1996,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_WRITE,COM3,8,
        55 55 00 00 00 00 00 00  | UU\#0\#0\#0\#0\#0\#0,
1997,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_WRITE,COM3,21,
        00 00 FF FF FF 00 0B F5 D4 40 01 07 08 09 0A 0B 0C 0D 0E 97 00  |
\#0\#0???\#0\#11??@\#1\#7\#8\#9\#10\#11\#12\#13\#14?\#0,
1998,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_READ,COM3,1,
        00  | \#0,
1999,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_READ,COM3,1,
        00  | \#0,
2000,14:58:00,HAL_Example_PN51x+PN53x.exe(4080),IRP_MJ_READ,COM3,4,
        FF 00 FF 00  | ?\#0?\#0,
2001,14:58:00,HAL_Example_PN51x+PN53x.exe(3896),IRP_MJ_READ,COM9,1,
        00  | \#0,
2002,14:58:00,HAL_Example_PN51x+PN53x.exe(3896),IRP_MJ_READ,COM9,1,
        00  | \#0,
2003,14:58:00,HAL_Example_PN51x+PN53x.exe(3896),IRP_MJ_READ,COM9,4,
        FF F3 0D D5  | ?ó\#13?,
2004,14:58:00,HAL_Example_PN51x+PN53x.exe(3896),IRP_MJ_READ,COM9,34,
        87 40 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15
16 17 18 19 1A 1B 1C 1D 1E 1F  |
?@\#0\#1\#2\#3\#4\#5\#6\#7\#8\#9\#10\#11\#12\#13\#14\#15\#16\#17\#18\#19□□□□□□□□□
,
2005,14:58:00,HAL_Example_PN51x+PN53x.exe(3896),IRP_MJ_READ,COM9,1,
        20  |  ,
2006,14:58:00,HAL_Example_PN51x+PN53x.exe(3896),IRP_MJ_READ,COM9,44,
                                   Page 3
```

Figure 4.3. Command data sniffed

Chapter 5

How to Use The PN532 Command

This chapter will introduce the details about how to use NFC module as peer-to-peer communication. And how host controller communicates with PN532, what the protocol is using during sending and receiving data. [30][29][31]

5.1 PN532 host link protocol

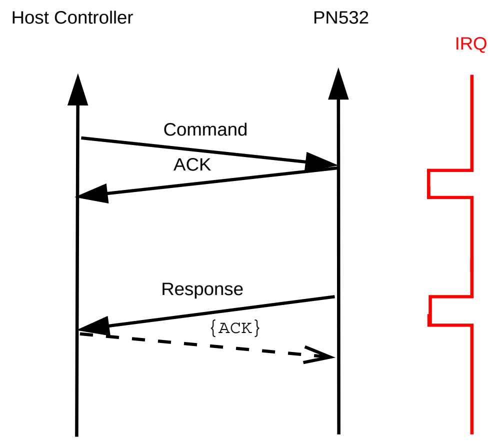


Figure 5.1. PN532 data flow

The host link protocol of PN532 is predefined by NXP Semiconductors. The basic exchanging command starts from passing messages: host controller sends command packet and gets ACK frame answered from PN532 as soon as packet is correctly received. And then the receiving messages: a response packet sent by PN532 and gets ACK from host(not necessary). At the same time polling mechanism or IRQ is used to ensure data exchanging successfully. Communication between the host controller and the PN532 is in a half-duplex mode which performed through frames. There has four different types of frames are used in one or both directions.

5.1.1 Normal and Extended Information Frame

There has two fixed command and response frame structures defined by NXP. They are the standard frame and extended frame. The standard frame can only send 255 bytes data with one frame. The extended frame can exchange more data between host and PN532 from 1 bytes to maximum theoretically up to 64kB(But the PN532 firmware limited the maximum length of the packet data to 264 bytes). Depends on the length of data, we use the suitable type of frame to transmit it. Because normal frame is similar with extended frame, we only introduce extended frame here.

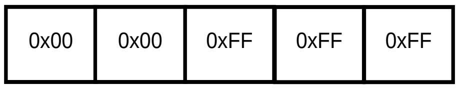


Figure 5.2. First 5 Fixed Bytes

In the extended information frame, the first five bytes of the frame are always fixed. It starts with a preamble bytes 0x00, then two bytes start of packet code 0x00 0xFF. The PN532 uses this synchronization pattern to indicate the beginning of a frame and tells all the previous data are ignored. The last two bytes means normal packet length and normal packet length checksum are 0xFF 0xFF.

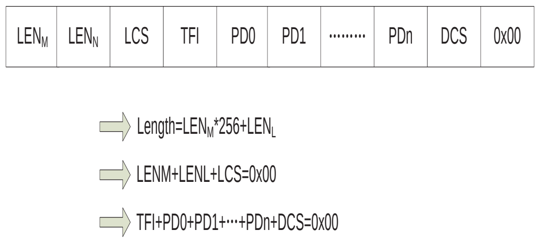


Figure 5.3. Other Bytes of The Frame

After there are two real packet length bytes and one real packet length checksum, then one byte the PN532 frame identifier TFI whose value depends on the way of the message(D4h or D5h). Followed is data bytes, the first byte of data is the command code. The later byte is packet data checksum. And the frame ends with a postamble bytes 0x00.

5.1.2 ACK Frame and NACK Frame

Both the specific ACK and NACK information frame is used for the synchronization of the packets. But ACK can be used either from the host controller to the PN532 or from the PN532 to the host controller, and indicates that the previous frame has been successfully received. NACK frame is only used from the host controller to the PN532 to indicate the failure of receiving. And NACK always leads a retransmission. Figure 6.4 shows the details figure of ACK frame and NACK frame.

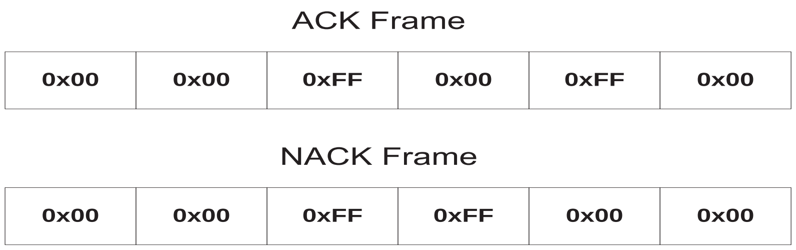


Figure 5.4. ACK and NACK Frames

5.1.3 Error Frame

The error frame is used to indicate the error at the application level. In the error frame, there is a specific application level error byte, which informs the host controller on the error reason of the command. When it is null, the operation has gone properly. The details of every error code meaning can be seen in the PN532 user manual.

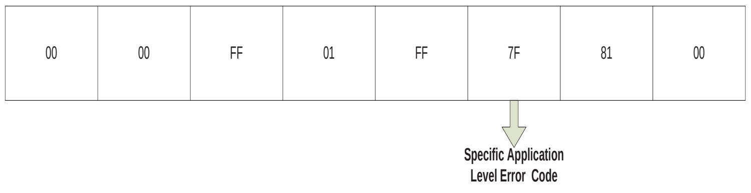


Figure 5.5. Error Frames

Chapter 6

Implement Application on VNC

In this chapter we connect the hardware, write code for the application(in different transmission BR, both active and passive data exchange protocol), test the code on PC, integrate code into VNC, optimize the code and validate the whole system in the end. Step by step, we write our own NXP NFC chip software and implement it into WSN system.

6.1 Connection of Hardware

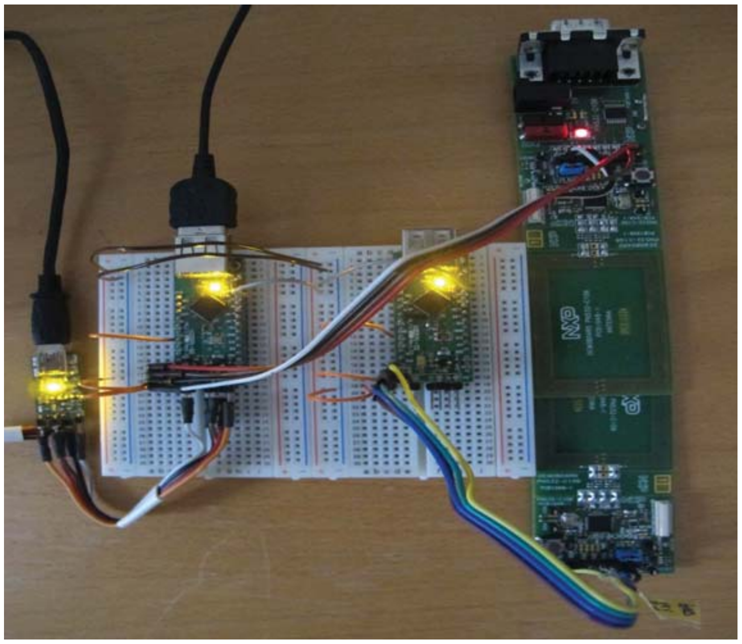


Figure 6.1. the Final Demo Hardware Connection

PN532 has three different connection interfaces to connect with the host controller. Inside the PN532, Host Interface module connects with the MCU 80C51, which uses firmware to process data to support NFCIP-1 protocol. Then 80C51 connects to the Contactless Interface Unit. Which connection interface is chosen depends on hardware configuration during the power up sequence of the chip. It can choose HSU(high speed UART) or I2C or SPI. The reference board is preconfigured in HSU mode and default serial speed is 115200 bauds. There has two pairs of UART pins on PCB1948-1. one pair is directly lead from PN532 pin 27(NSS) and pin 28(MOSI), so its working voltage is 3.3V(CMOS). Another pair is the first pair go through a CMOS to RS232 voltage converter chip–ADM3202ARUZ, which makes it supporting RS232 working voltage. We need connect PN532 UART directly to the VNC, so we use the first pair of pins. on another part, VCN connects with tablet using mini USB adapter. We can see hardware connection block on figure 6.1.

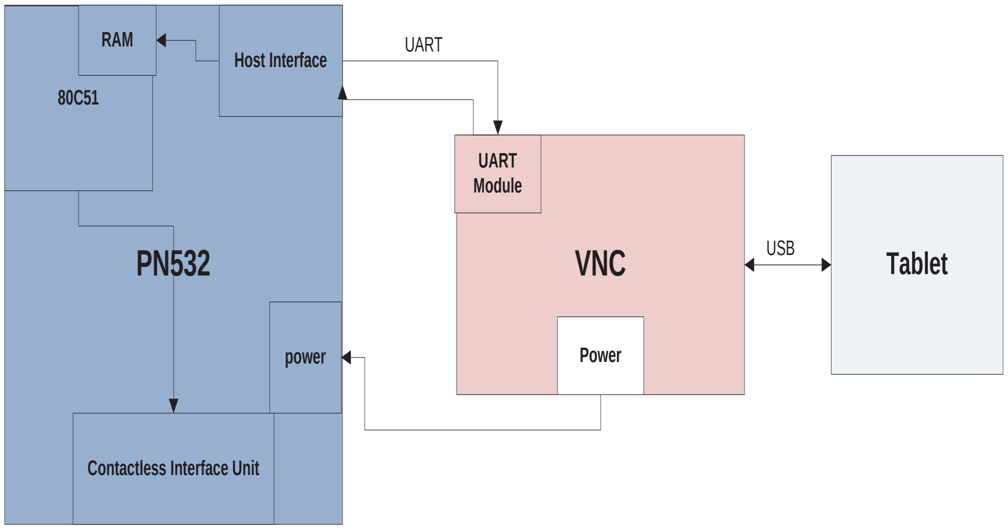


Figure 6.2. Schema Hardware Connection

6.1.1 Additional Lines–IRQ(DTR) and HREQ(DSR)

There has two additional lines available, once the interface with the host controller is chosen. IRQ informs the host controller to know when a response from PN532 is ready. Handshake mode must be set to use one or two additional lines. The reference board is always set in handshake mode, so it means we can use additional lines as we want. Using UART flow control has a lot of advantages, speeds up communication and reduces the bus overall traffic. Host controller can know when is ready to send its frame by using the IRQ signal. Also there is no need for host controller always reads the status byte, which will reduce the overall traffic on the UART.

However most of the sensor nodes has not UART DTR/DSR pins, they only have

Tx and Rx two pins. Moreover iPack Center WSN system has not UART flow control pins. So we don't use additional lines in this thesis project.

6.2 Application Code

After learning the sniffer results, HAL Port Manual and PN532 user manual, we begin to write own application code. Although my partner and I are working on VNC firmware code at the same time. we choose write application code in MS Visual Studio first. Because Windows is more stable than our own developed firmware. We use PC RS232 COM port to send and receive command to/from PN532 to ensure application code is right. The application is supposed to use in wireless sensor network, so power consumption is very important. The application will use low power mode of PN532.

6.2.1 Low Power mode and Wake up

The PN532 has different low power modes for both contactless interface and CPU that can be choose.

Power modes for Contactless interface

The contactless interface can divided into two parts, the analog front-end part and contactless UART part. There has totally five low power modes for them.
1: Hard Power Down mode. Both parts are in reset state. And this mode cannot be reached by a firmware action.
2: CL_A mode. The contactless UART is running, the analog front end is operational and the RF field is not generated. This thesis project use this low power mode when setting target/PICC mode.
3: CL_B mode. Only difference of CL_A is it generates RF field. The PN532 as initiator/PCD communication mode, using this mode to save power consumption.
4: CL_C mode. In this mode, the contactless UART is in power down mode, the analog front end is partially operational(only the RF level detector is active) and RF field is not generated. LowVbat, target/PICC and virtual card mode can use this low power mode.
5: CL_D mode. Both parts are set to the minimum power consumption mode. Even RF level detector is not activated. This mode is used in stand by mode with no session of lowVbat, initiator, target, or virtual card mode.

Power modes for CPU

Besides PN532 proved several power modes for CPU.
1: The first one is hard power down mode, in this mode CPU is in reset state and doesn't work. This mode cannot reach by a firmware action, only can be control by an external action on RSTPDN pin. This mode need manually reach, so we don't

consider it.
2: The second mode is normal mode, which CPU never stop running. It cost too much power consumption under this mode.
3: The last mode is power down mode, when PN532 works in this model, oscillator is stopped when it is idle.
PN532 works under low power mode likes a wheel we can see in figure 6.2. It has four state: Idle, Initialize, Listen and Data Communication. When it doesn't work it always in IDLE power down mode, once start one phase works after another, it will not stop until user stop or remote device, after that it will go back to idle power down phase.

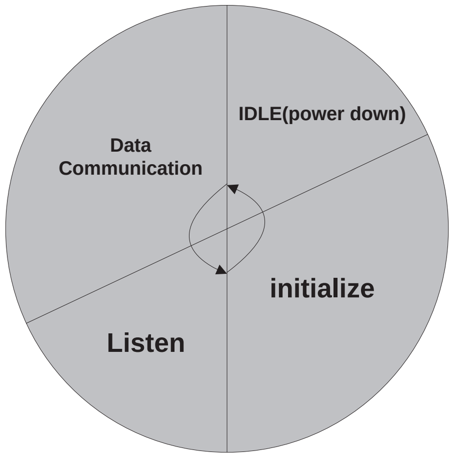


Figure 6.3. PN532 Low Power working state

Wake Up

Low power mode can save power consumption, increase sensor nodes life time. As the cost, it takes 50 ms to wake up the PN532 from sleeping. In order to exit from Power Down mode, the host controller need to send a command to the PN532 on th HSU link. Besides to avoid the command will not be lost or partially received, either we send a large preamble containing dummy data command or a 0x55 dummy byte and wait for the waking up delay before sending the command frame. In this thesis project, we use 0x55 0x55 0x00 0x00 0x00 0x00 0x00 0xFF 0x03 0xFD 0xD4 0x14 0x01 0x17 0x00 command to wake up the PN532. Here SAMConfiguration command D4 14 which is used to select the data flow path by configuring the internal serial data switch. D4 14 01 means the SAM is not used and it works under normal mode. And IRQ byte is not present, it indicates PN532 ignoring the IRQ pin.

6.2.2 Baud Rate Setting

After wake up, both initiator and target need to set their working baud rate first. The PN532 HSU is up to 1288 kbauds. We set HSU speed with SetSerialBaudrate command. If host controller wants to change the baudrate after set, an ACK frame must be send to the PN532 after reception of SetSerialBaudrate response. The PN532 will switch to the new baudrate later. In this project, after considering both speed and power factors, we use 115200 bauds.

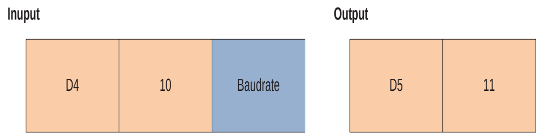


Figure 6.4. SetSerialBaudrate Command

6.2.3 As a NFC Target setting

One of the PN532 is set to target in a NFC peer-to-peer communication. First the host controller need to use WriteRegister command to overwrite content of XRAM memory inside PN532. Though we learn to use this command from sniffing HAL software, we don't know why this command is used in this target setting sequence. The register address 0x02F9 changed by this command cannot be found in all the NXP Semiconductors or 80C51 technical manual. So we just use the same command as HAL software use in this project application.

Then the application using TgInitAsTarget command to set the PN532 as target. The command frame can be seen in figure 6.4. The mode byte is set to 0x02 which means the target should respect DEP (data exchange protocol)only, and supports both active and passive requirement. Because the application only communicate as peer-to-peer mode, so the MifareParams and FeliCaParams bytes can be any hex numbers. NFIC3t is used in the ATR_RES(Attribute Response) in case of ATR_REQ(Attribute Request) received from the initiator. The application set NFIC3t to 0x0A, So initiator should set its ATR_REQ to 0x0A bytes length. The LEN Gt byte indicates the length of ATR_RES which is set to 0x0C. So Gt is the content of ATR_RES. It is set the HEX format "I am from NXP" her. Later byte LEN Tk is set to 0x00, so there is no content in Tk array. After this command, the PN532 has configured as target and will enter in power down mode until external initiator RF field wakes it up.

6.2.4 As a NFC Initiator setting

Another PN532 in the thesis project is set to Initiator. Initiator configuration basically consists several RF communication command. First command is RF-Configuration which can been seen in figure 7.5. The thesis application only set

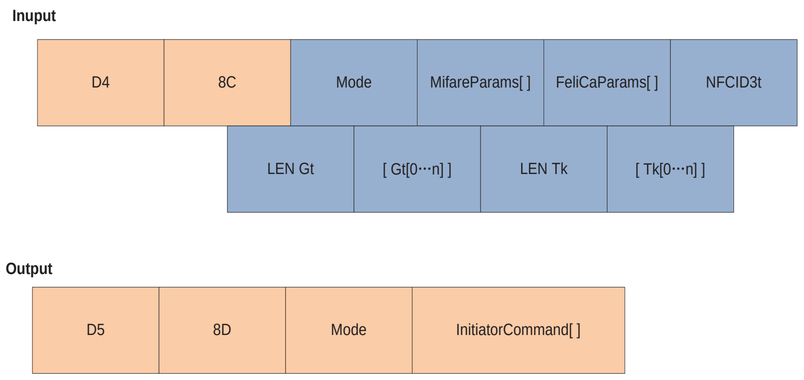


Figure 6.5. Set Target Command

cfgItem01(RF field) for the PN532. The ConfigurationData is set to 0x02 which means Auto RFCA(RF Collision Avoidance) is on and RF is off. That means the PN532 initiator will not switch on its own RF field before detecting external field which will save power consumption before detecting target.

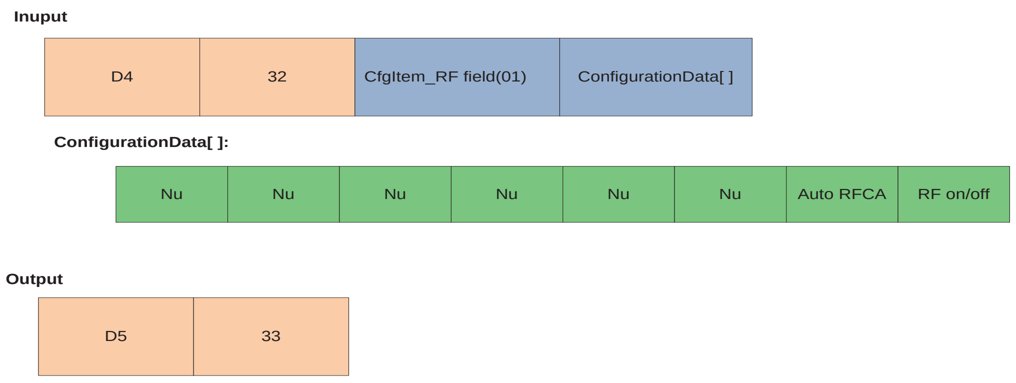


Figure 6.6. RF Configuration Command

After followed with the command SetParameters which is used to set the PN532 internal parameters and its behaviour. The detailed command frame can be seen on figure 7.6. In this thesis project, the internal flag setting is just a little difference form the default one, which is only ATR_RES is used during communication. Because other bit is for ISO14443 protocol and useless in NFCIP-1 protocol p2p communication.

Before beginning Peer-to-Peer communication, there still lefts one command to configure the initiator. Host controller uses InJumpForDEP command to active a target whatever active or passive mode it is. This command contains passive/active mode, baud rate, initiator random id and optional general information bytes. Next is set to 0x05, and it informs that PassiveInitiatorData is present in the command frame and Gi is present in the command frame. NFCID3i is the content

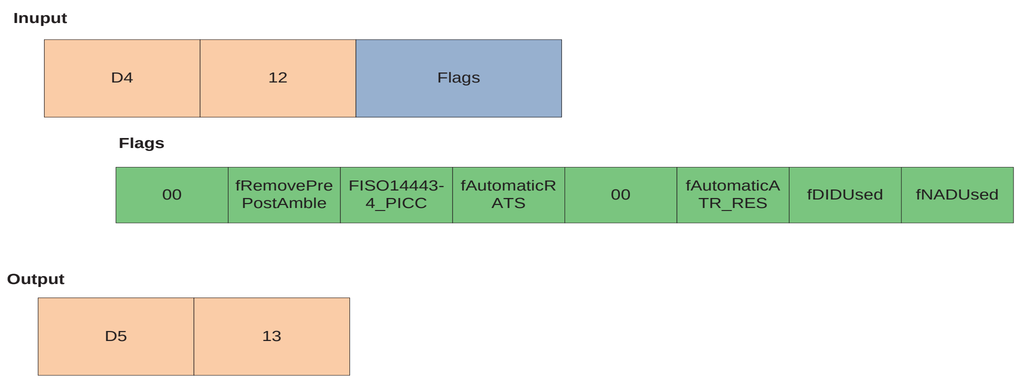


Figure 6.7. Initiator SetParameters Command

of ATR_REQ, and the application set it to the HEX format "Who are you". So the initiator will receive ATR_RES "I am from NXP" after Target receiving ATR_REQ "Who are you".

After this command sending from host controller to the PN532, the NFC initiator RF field checks whether there has target near by and will get ATRRES sent by any target in the RF field. ATRRES frame contains parameters of NFC target configuration(the most important information is the target Tg), in case of initiator will communicate with it. Also initiator send ATR_REQ frame to target, letting it knows there has initiator in field, in order to get ready to be waked up for peer-to-peer communication.

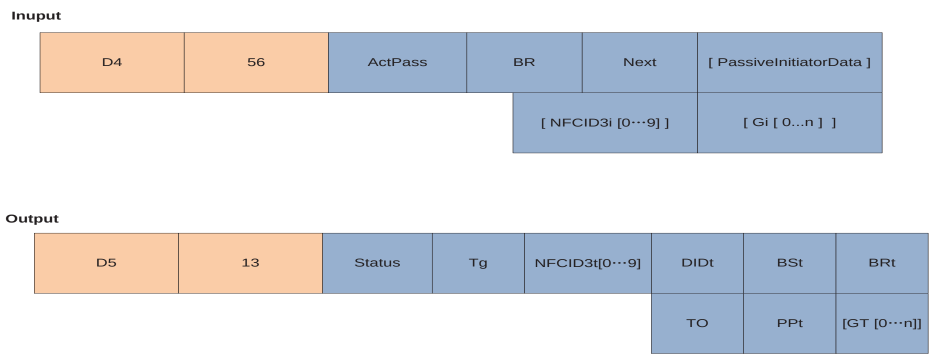


Figure 6.8. Initiator InJumpForDEP Command

6.2.5 Data Exchange

With both NFC initiator and target in RF field are set to Data Exchange Protocol (DEP) and matches. The initiator can exchange data with the target now. In case of there has more than one configured target in the RF field, the host controller sends command InSelect to the initiator to select a specific target in RF field to communicate with. The Tg byte in command is the logical number of the target, and it is sent to initiator before by target response ATRRES frame.

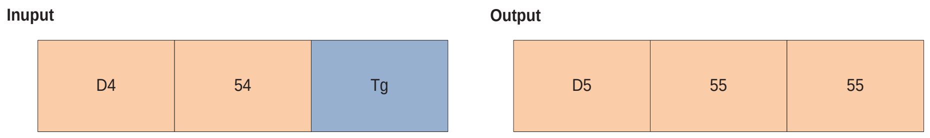


Figure 6.9. Initiator InSelect Command

After selecting status response correctly, peer-to-peer begins. The protocol of data exchanges between the initiator and the target are depend on all the configurations stored internally by the PN532 before. The order of communication is the initiator sends data to the target first, then the target sets data and the initiator receives data. If the total length of data is more than 264 bytes which is the PN532 firmware limtied maximum length of the packet data, the above exchanging flow should iterate until all the data transmission is finish. There has three main command involved in the above exchanging flow, InDataExchange, TgGetData and TgSetData. The initiator PN532 use InDataExchange command sends its dataout[] array, the target host controller sends TgGetData command to receive the dataout[] array. Then the target uses TgSetData transmits its datain[] array to the initiator, and the initiator gets it from the response frame of InDataExchange.

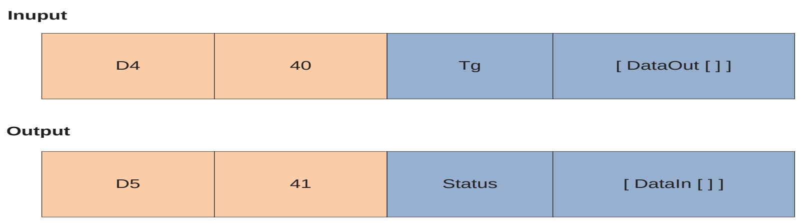


Figure 6.10. Initiator InDataExchange Command

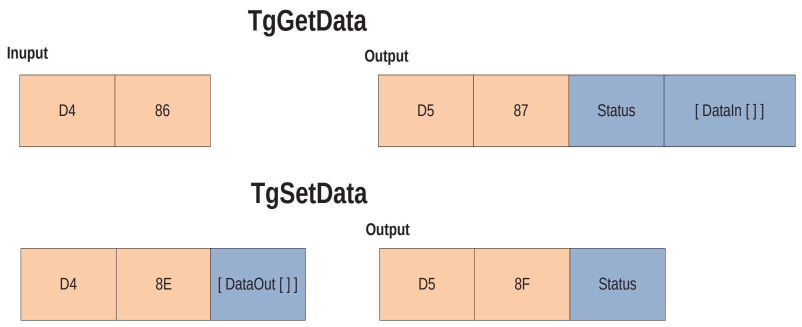


Figure 6.11. Target TgGetData & TgSetData Command

The data exchanging procedure iterates several times until all the data has been transmitted, the peer-to-peer communication is finished then. The initiator host controller uses InDeselect command D4 44 Tg to deselect the target. After that the released target PN532 enters in power down mode again for power save.

6.2.6 Overview of our own developing NXP PN532 SDK

All above command code are written and debug in Microsoft Visual Studio 2010. It releases two console applications, one for the initiator and another for the target. The applications are in charge of using PC RS232 to send commands to the PN532 in order to initialize and control them doing the Peer-to-Peer communication. These two console applications are successfully controlling the PN532 doing different baud rates and different DEP (Data Exchange Protocol) data exchanging. So the code can be used as our own SDK to design and develop NXP NFC chip applications on any platforms with only very little changes. As Figure 6.11 shows overview flow of the PN532 doing the Peer-to-Peer communication.

6.3 SDK Integrated Into VNC

We figured out how to use the PN532 as initiator and target, in addition we have developed and validated our own easy integrated SDK. Now we need make the NFC enable WSN system by integrating SDK into VNC.

6.3.1 VNC firmware code

First we need to write our own VNC firmware code which should support both UART write and read, also maybe some GPIO for debug using. The firmware project in thesis project contains six source code file, and three of them are header files, others are C code files.[32]

bb _ iomux.c

New project wizard is using to auto generate this file which function is set up a project. This is an auto-generated file while using new project wizard to setup a project. It provides a correspondence between I/O ports and package pins. At most five connection interface are designed on 32 pins Vinculum II chip module to use in the firmware. But in this thesis project, we just use one USB interface, one UART interface, and one GPIO interface.

For the UART interface, sensor nodes normally only have TX and RX pins, so we only configured these two basic pins and ignore other flow control pins. In addition, the UART TX is mapped to pin 29 and UART RX signal is mapped to pin 30.

Because of limited pins number, there has only two GPIO pins are used in the firmware configuration. One is mapped to pin 31 and another is mapped to the pin

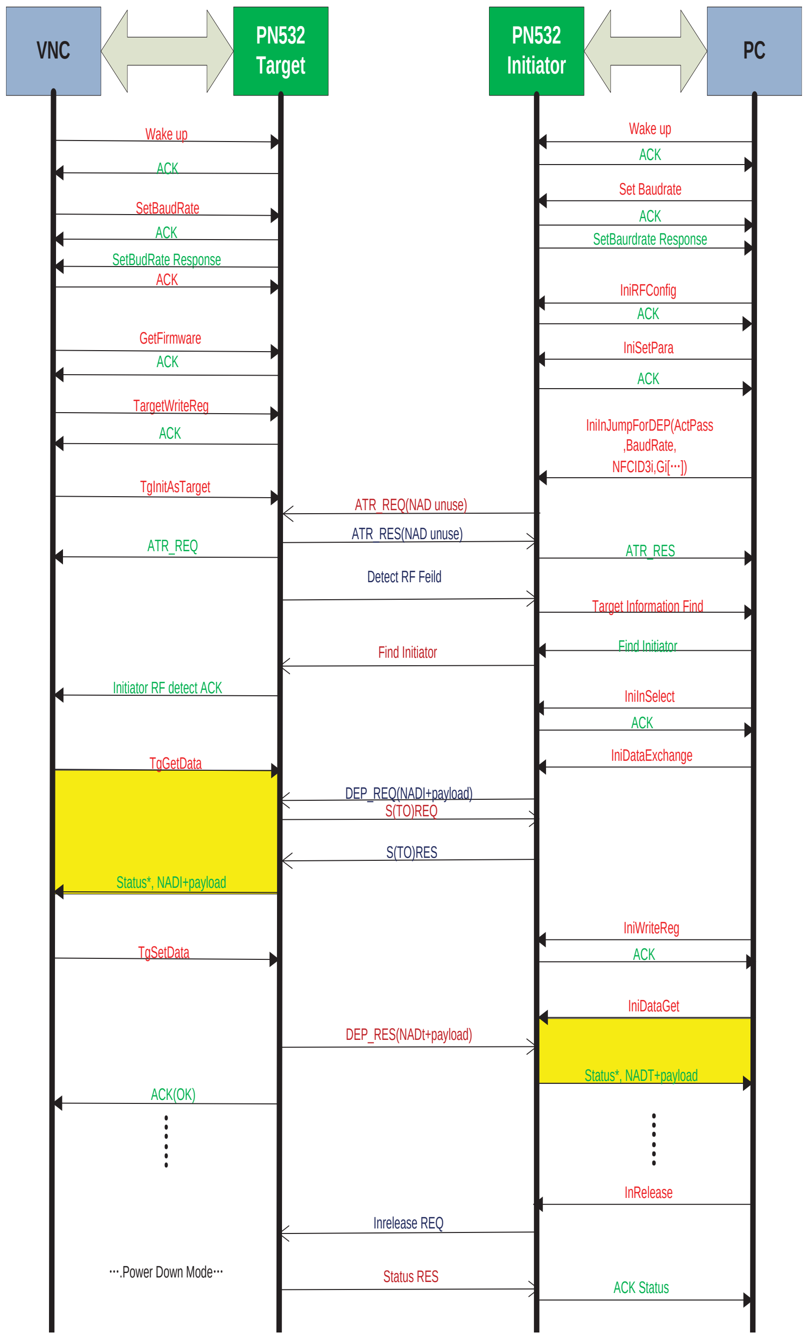


Figure 6.12. Overview flow of The PN532 P2P Communication SDK

15. GPIO interface is not a part of the NFC enable WSN system, but it is used to send some signals for debugging and testing purports.

bb.h & bb.c

These two files are code for the RTOS(real time operating system). The bb.h file is included all the used libraries which contain port drivers, such as USBHOST.h. The bb.c file contains the program entry function.

There has five tasks in the entry function at all. The first task is setup the execution environment, the second one is to allocate the buff for each interface. Then initialize the USB interface task which must be done before other interface initialization. After the CPU create the threads and semaphores. When everything above have finished, the last task initialize other used interface.

ZnParam.h, znFunc.h & znFunc.c

The ZnParam.h contains all predefined parameters of the system. Such as Clock speed and UART baud rate.

The znFunc.h file contains all the function declaration used in znFunc.c. While there has three thread implementation functions and read/write ports functions in znFunc.c. The most used read/write function is for UART interface. We have UART_RD, UART_WR function which can automatically check the end byte of the buffer or send the '0' flowing the end of content. Furthermore there also has two function UART_CMD_RD/WR (char) which need to specify the length of the buffer in case of there has '0' byte in transmitting data. With these four UART interface controlling functions, we can easily integrated our PC command into VNC. Also the GPIO port has its own read/write functions which are called GPIO_RD(void) and GPIO_WR(int value). These two function is for led connect to the GPIO or just send signal to GPIO port for testing.

The Procedure of the Firmware

Basically the firmware on VNC2 is a priority driven RTOS. Every threads should have their own stack and priority. For both sensor node and NFC adapter using, we create three threads. They are USB connection checking thread, UART read and write thread and USB Transmission thread. Considering the system requirement, USB connection has the highest priority but the highest stack size. The UART read and write thread has almost same size stack of USB detecting stack, and weight the second priority. Also USB Transmission thread has the lowest priority of them.

When the schedule starts, the kernel is launched. With a short delay, these three threads are blocked and wait for their semaphore signals. Because the USB connection checking thread has the highest priority, it almost get its semaphore and runs. The USB connection checking thread is a dead loop before it detect the connection is right, after USB connects ready, it will set flag to 1 and release the semaphore to other threads. After that it will has a 100 ms period sleeping time.

Other two threads almost do the same work, getting the semaphore, reading and writing from/to ports, releasing the semaphore and has a 5ms sleep period. Figure 6.12 shows the example execution of firmware RTOS procedure.

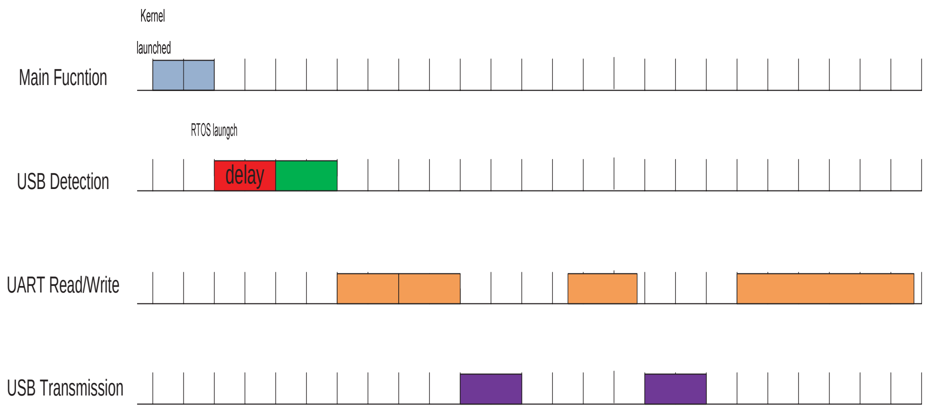


Figure 6.13. the Firmware RTOS procedure

6.3.2 Integrated

As soon as the VNC2 firmware has developed already, we begin to integrated our own design SDK into it. At first we need to modify the SDK for the new hardware environment. Because the SDK is developed on PC under Windows x86 operation system, the hardware and UART controller functions are different from VNC2. However integration is not difficult at all with our own design SDK. There is no need for us to consider of too much about hardware dependent drivers or rewriting a lot of code. We only need replace the UART read/write functions under Windows with VNC2 functions. Then the SDK is integrated into the VNC2 and will works correctly.

Both the initiator and target flow chart can be seen on figure 6.13 and figure 6.14. When VNC2 is power on, USB detection thread starts first. Then UART Read/Write thread begins, both the initiator and the target begin their initialization sequence. After initialization is finish, the target enters in software power down mode until the outside initiator RF is detected by its RF detector module. While the initiator detects and selects the target in its RF field. The first peer-to-peer data exchange is launched by the initiator, it sends data "Hi! Who are you" to the target. And when the target receives this data it sends back data "I am iPack Sensor Node" to the initiator. Just like the first time when the initiator get data from the target, It begins to send a ECG packet to the target. However the ECG packet length exceeds the maximum length of data the PN532 can transmit. So the whole ECG packet is divided to two parts to send to the target. And when target gets ECG packet, it send empty data packet back to the initiator. This peer-to-peer

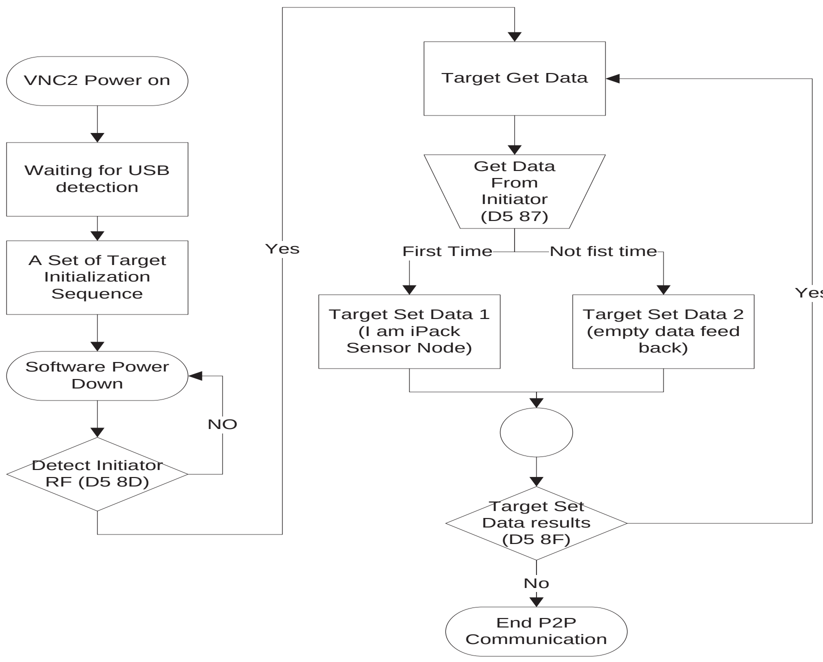

VNC2 Power on
Waiting for USB detection
A Set of Target Initialization Sequence
Software Power Down
NO
Detect Initiator RF (D5 8D)
Yes
Target Get Data
Get Data From Initiator (D5 87)
First Time
Not fist time
Target Set Data 1 (I am iPack Sensor Node)
Target Set Data 2 (empty data feed back)
Yes
Target Set Data results (D5 8F)
No
End P2P Communication

Figure 6.14. the Target Flow Chart

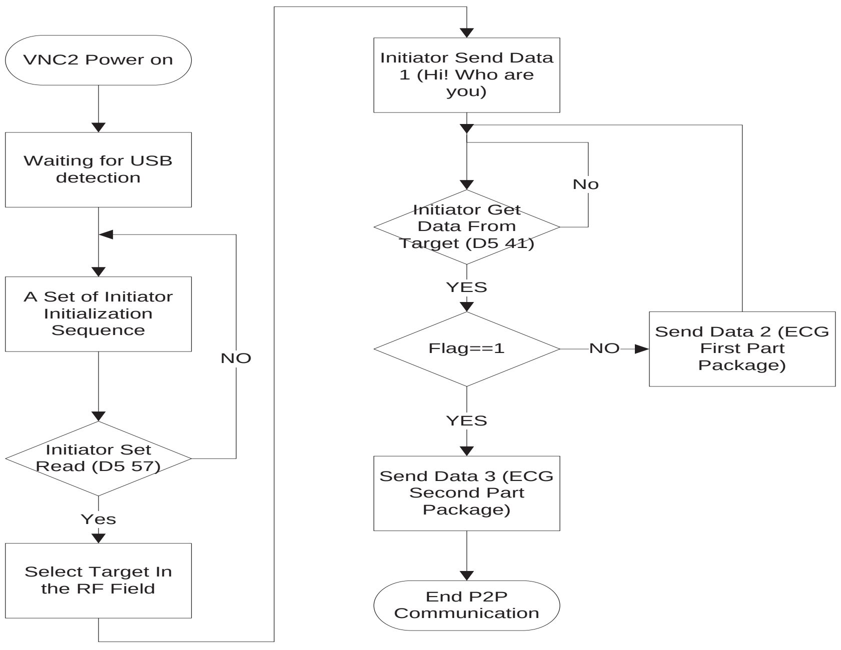

VNC2 Power on
Waiting for USB detection
A Set of Initiator Initialization Sequence
NO
Initiator Set Read (D5 57)
Yes
Select Target In the RF Field
Initiator Send Data 1 (Hi! Who are you)
No
Initiator Get Data From Target (D5 41)
YES
Flag==1
NO
Send Data 2 (ECG First Part Package)
YES
Send Data 3 (ECG Second Part Package)
End P2P Communication

Figure 6.15. the Initiator Flow Chart communication will repeat forever until the target is removed out of the initiator RF field. Then the communication ends.

6.3.3 testing and optimize

After integrated to VNC2, we do some testing about the stability and performance of the NFC adapter and the sensor node based on VNC2.

For the stability, both VCN2 IDE debugging tools and sniffer are used. When we monitor the NFC enable WSN system running, we find sometimes Peer-to-Peer communication is failed. We analyse the phenomenon by the debugging tools and find the initiator thread or target thread stuck on at the beginning. So we check the stack analyse tool, find stack for initiator or target is out of bounds. And VNC firmware is RTOS, when stack is not enough for the thread, this thread will never run and cause the system dead loop. We try to increase the stack for UART read/write thread and decrease both USB detection and USB Transmission threads to the minimum. However the stack is still not enough sometimes, and we cannot increase stack for UART read/write thread any more. Only method left is modify the SDK to decrease the stack needed. The original designed SDK offers different baud rate, active/passive and transmitting data self analysis which need a lot of statement control flow functions. And these control flow cost amounts of software stack. By modifying the code, we only use passive 424kbps transmitting speed with no data self analysis functions for our final NFC enable WSN system based on PN532 and VCN2. After that, the system runs stable.

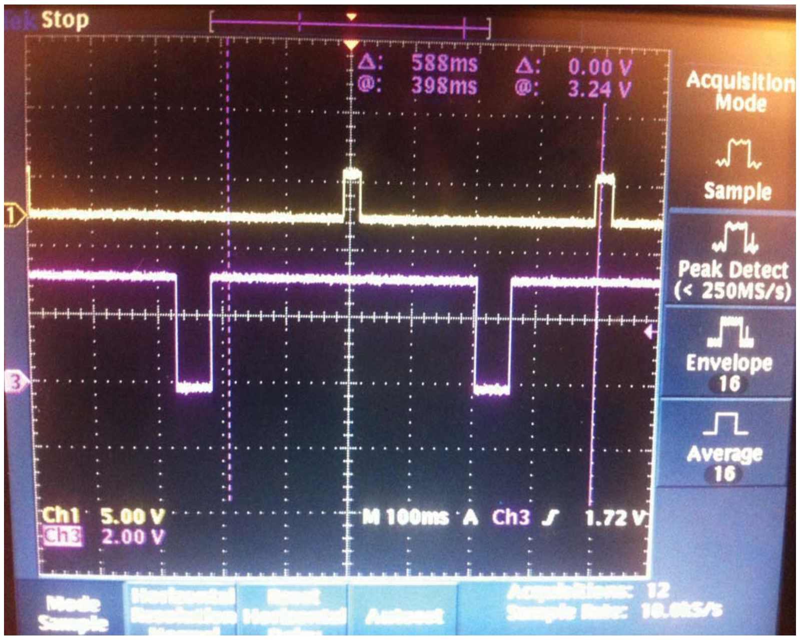
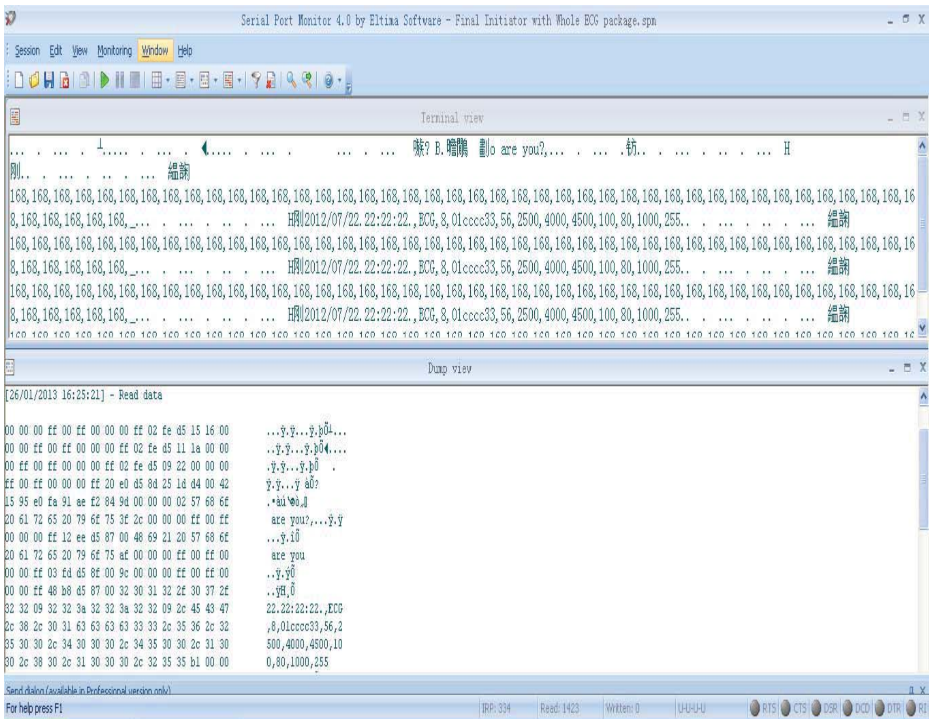


Figure 6.16. NFC enable WSN System P2P Communication Delay Testing

For the performance, we use oscilloscope to test the delay of the whole system when it is doing the peer-to-peer communication. The data exchanging begins when the initiator sends data to the target, and ends when initiator get data from the target. So at fist we set the initiator GPIO pin 15 signal to 1, and pin 31 to 0. When initiator begins sending data to the target, we set pin 15 signal to 0. After data sending completely, pin 15 is set to 1 again. Also pin 31 is set to 1 when initiator getting data from the target, and set to 0 when data is received completely. The oscilloscope testing result can be seen on figure 1.12. From the figure 6.13, we can find there has almost 250 ms delay of the transmitting. The delay is kind of larger than we though because of not using IRQ signals in UART read/write. Also there is still some statement flow control in UART read function, which leads delay of the system.

6.3.4 Final Demonstration


Serial Port Monitor 4.0 by Eltima Software - Final Initiator with Whole ECG package.spm
Session Edit View Monitoring Window Help
Terminal view
Dump view
Send dialog (available in Professional version only)
For help press F1
IRP: 334
Read: 1423
Written: 0
U-U-U-U
RTS CTS DSR DCD DTR RI

```
... . ... . ┴..... . ... . ◀.... . ... .      ... . ...   嗛? B.暗鸊  劃o are you?,... . ... .钪.. . ... . .. . ... H
刚.. . ... . .. . ...  縕謝
168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,16
8,168,168,168,168,168,_... . ... . .. . ...  H刚2012/07/22.22:22:22.,ECG,8,01cccc33,56,2500,4000,4500,100,80,1000,255.. . ... . .. . ...  縕謝
168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,16
8,168,168,168,168,168,_... . ... . .. . ...  H刚2012/07/22.22:22:22.,ECG,8,01cccc33,56,2500,4000,4500,100,80,1000,255.. . ... . .. . ...  縕謝
168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,168,16
8,168,168,168,168,168,_... . ... . .. . ...  H刚2012/07/22.22:22:22.,ECG,8,01cccc33,56,2500,4000,4500,100,80,1000,255.. . ... . .. . ...  縕謝
```

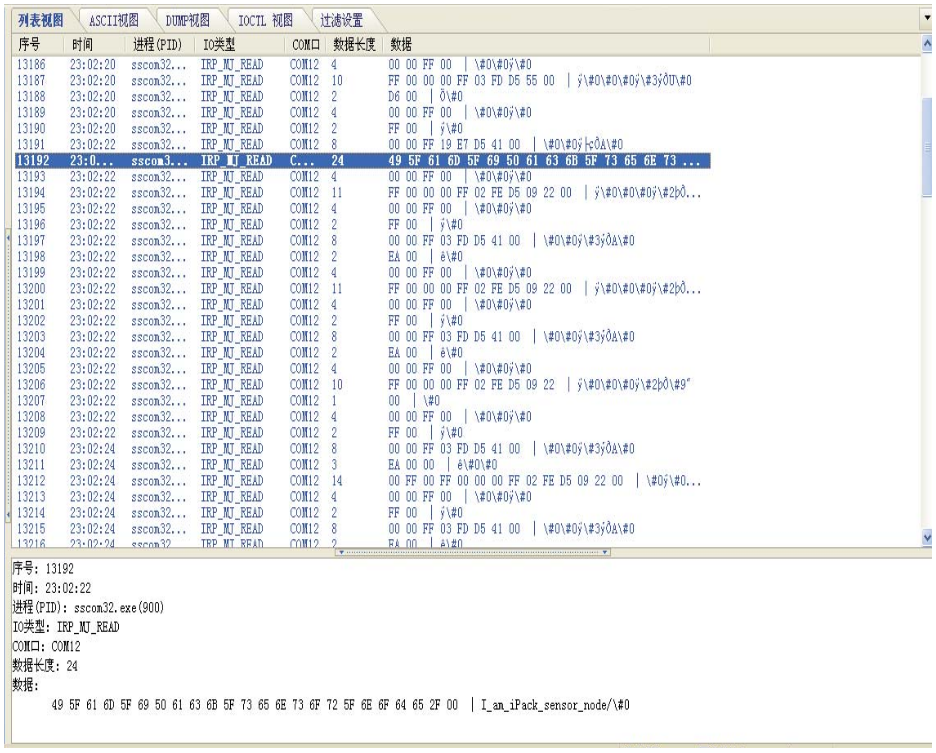
```
[26/01/2013 16:25:21] - Read data

00 00 00 ff 00 ff 00 00 00 ff 02 fe d5 15 16 00     ...ÿ.ÿ...ÿ.þÕ┴...
00 00 ff 00 ff 00 00 00 ff 02 fe d5 11 1a 00 00     ..ÿ.ÿ...ÿ.þÕ◀....
00 ff 00 ff 00 00 00 ff 02 fe d5 09 22 00 00 00     .ÿ.ÿ...ÿ.þÕ   .
ff 00 ff 00 00 00 ff 20 e0 d5 8d 25 1d d4 00 42     ÿ.ÿ...ÿ àÕ?
15 95 e0 fa 91 ae f2 84 9d 00 00 00 02 57 68 6f     .•àú'®ò„▯
20 61 72 65 20 79 6f 75 3f 2c 00 00 00 ff 00 ff      are you?,...ÿ.ÿ
00 00 00 ff 12 ee d5 87 00 48 69 21 20 57 68 6f     ...ÿ.îÕ
20 61 72 65 20 79 6f 75 af 00 00 00 ff 00 ff 00      are you
00 00 ff 03 fd d5 8f 00 9c 00 00 00 ff 00 ff 00     ..ÿ.ýÕ
00 00 ff 48 b8 d5 87 00 32 30 31 32 2f 30 37 2f     ..ÿH¸Õ
32 32 09 32 32 3a 32 32 3a 32 32 09 2c 45 43 47     22.22:22:22.,ECG
2c 38 2c 30 31 63 63 63 63 33 33 2c 35 36 2c 32     ,8,01cccc33,56,2
35 30 30 2c 34 30 30 30 2c 34 35 30 30 2c 31 30     500,4000,4500,10
30 2c 38 30 2c 31 30 30 30 2c 32 35 35 b1 00 00     0,80,1000,255
```

Figure 6.17. the Final Demo Target Sniffer Results

Figure 6.17 is the final demo target peer-to-peer communication sniffer results. It is very clearly the whole ECG packet is divided to parts, receiving by the target repeatedly. Due to the reason that sniffer software doesn't supports HEX to ASCII very well, we can not see the whole first time receiving data in ASCII. It displays

" are you? " in both terminal and dump view. But look at the dump view, just in the end of last row of " are you? " has HEX 0x57 0x68 0x6F which is ASCII "Who".



```
列表视图  ASCII视图  DUMP视图  IOCTL 视图  过滤设置
序号    时间      进程(PID)   IO类型       COM口  数据长度  数据
13186   23:02:20  sscom32...  IRP_MJ_READ  COM12  4    00 00 FF 00  | \#0\#0ÿ\#0
13187   23:02:20  sscom32...  IRP_MJ_READ  COM12  10   FF 00 00 00 FF 03 FD D5 55 00  | ÿ\#0\#0\#0ÿ\#3ýÕU\#0
13188   23:02:20  sscom32...  IRP_MJ_READ  COM12  2    D6 00  | Ö\#0
13189   23:02:20  sscom32...  IRP_MJ_READ  COM12  4    00 00 FF 00  | \#0\#0ÿ\#0
13190   23:02:20  sscom32...  IRP_MJ_READ  COM12  2    FF 00  | ÿ\#0
13191   23:02:22  sscom32...  IRP_MJ_READ  COM12  8    00 00 FF 19 E7 D5 41 00  | \#0\#0ÿ├çÕA\#0
13192   23:0...   sscom3...   IRP_MJ_READ  C...   24   49 5F 61 6D 5F 69 50 61 63 6B 5F 73 65 6E 73 ...
13193   23:02:22  sscom32...  IRP_MJ_READ  COM12  4    00 00 FF 00  | \#0\#0ÿ\#0
13194   23:02:22  sscom32...  IRP_MJ_READ  COM12  11   FF 00 00 00 FF 02 FE D5 09 22 00  | ÿ\#0\#0\#0ÿ\#2þÕ...
13195   23:02:22  sscom32...  IRP_MJ_READ  COM12  4    00 00 FF 00  | \#0\#0ÿ\#0
13196   23:02:22  sscom32...  IRP_MJ_READ  COM12  2    FF 00  | ÿ\#0
13197   23:02:22  sscom32...  IRP_MJ_READ  COM12  8    00 00 FF 03 FD D5 41 00  | \#0\#0ÿ\#3ýÕA\#0
13198   23:02:22  sscom32...  IRP_MJ_READ  COM12  2    EA 00  | ê\#0
13199   23:02:22  sscom32...  IRP_MJ_READ  COM12  4    00 00 FF 00  | \#0\#0ÿ\#0
13200   23:02:22  sscom32...  IRP_MJ_READ  COM12  11   FF 00 00 00 FF 02 FE D5 09 22 00  | ÿ\#0\#0\#0ÿ\#2þÕ...
13201   23:02:22  sscom32...  IRP_MJ_READ  COM12  4    00 00 FF 00  | \#0\#0ÿ\#0
13202   23:02:22  sscom32...  IRP_MJ_READ  COM12  2    FF 00  | ÿ\#0
13203   23:02:22  sscom32...  IRP_MJ_READ  COM12  8    00 00 FF 03 FD D5 41 00  | \#0\#0ÿ\#3ýÕA\#0
13204   23:02:22  sscom32...  IRP_MJ_READ  COM12  2    EA 00  | ê\#0
13205   23:02:22  sscom32...  IRP_MJ_READ  COM12  4    00 00 FF 00  | \#0\#0ÿ\#0
13206   23:02:22  sscom32...  IRP_MJ_READ  COM12  10   FF 00 00 00 FF 02 FE D5 09 22  | ÿ\#0\#0\#0ÿ\#2þÕ\#9"
13207   23:02:22  sscom32...  IRP_MJ_READ  COM12  1    00  | \#0
13208   23:02:22  sscom32...  IRP_MJ_READ  COM12  4    00 00 FF 00  | \#0\#0ÿ\#0
13209   23:02:22  sscom32...  IRP_MJ_READ  COM12  2    FF 00  | ÿ\#0
13210   23:02:24  sscom32...  IRP_MJ_READ  COM12  8    00 00 FF 03 FD D5 41 00  | \#0\#0ÿ\#3ýÕA\#0
13211   23:02:24  sscom32...  IRP_MJ_READ  COM12  3    EA 00 00  | ê\#0\#0
13212   23:02:24  sscom32...  IRP_MJ_READ  COM12  14   00 FF 00 FF 00 00 00 FF 02 FE D5 09 22 00  | \#0ÿ\#0...
13213   23:02:24  sscom32...  IRP_MJ_READ  COM12  4    00 00 FF 00  | \#0\#0ÿ\#0
13214   23:02:24  sscom32...  IRP_MJ_READ  COM12  2    FF 00  | ÿ\#0
13215   23:02:24  sscom32...  IRP_MJ_READ  COM12  8    00 00 FF 03 FD D5 41 00  | \#0\#0ÿ\#3ýÕA\#0
13216   23:02:24  sscom32     IRP_MJ_READ  COM12  2    EA 00  | ê\#0

序号: 13192
时间: 23:02:22
进程(PID): sscom32.exe(900)
IO类型: IRP_MJ_READ
COM口: COM12
数据长度: 24
数据:
      49 5F 61 6D 5F 69 50 61 63 6B 5F 73 65 6E 73 6F 72 5F 6E 6F 64 65 2F 00  | I_am_iPack_sensor_node/\#0
```

Figure 6.18. the Final Demo Initiator Sniffer Results

Figure 6.18 shows the final demo initiator peer-to-peer communication sniffer results. The HEX 0xD541 means data receiving from the target. We can see the initiator received the data from target was "I_am_iPack_sensor_node" in ASCII format. And other data was empty packet.

Part III

Conclusion and Future Work

Chapter 7

Conclusion

Although there were some small changes compared to the thesis proposal, the thesis project was still successful, because of the NFC enable WSN system demo finished in the end.

7.1 Overview of The Thesis Work

The whole thesis project is consist of handling out a basic idea, doing some research, choosing hardware and building software environment, designing NFC adapter hardware connection, study the PN532 command, Writing own SDK, writing VNC2 firmware, Integrating SDK and optimizing the code. We make a demo of NFC enable WSN system demo which can transmit ECG packet with peer-to-peer communication and data rate speed is up to 424kbps passive/active in the end of the project.

There had two big changes compared to the project proposal. The first one was we couldn't use the sensor node PCB SSN-v3.8 due to no UART TX or RX pin is on the PCB board. It was a little pity that cannot use this thesis demo directly into iPack Center existing WSN system. However I spent a lot of time on learning the ultra-low power MCU MSP430, which is kind of interesting. The second one was we couldn't write the PN532 controlling code into 80C51, because there was no debug port for 80C51 on the PN532. Also no command allowed us fully access to the RAM area in the 80C51. So we just added another MCU VNC2 to the project for programming on it. Although there has many difficulty during my thesis work, we overcome them and makes thesis successful in the end.

For using the VNC2 chip in the project thesis, I had to cooperate with a partner. we all needed to use the VNC as our project demo platform. Furthermore we developed a medicine box project together. I integrated this NFC adapter and a small RFID application into the medicine box. The NFC adapter is used to receive patient informations from outside NFC-enable mobile or card and the RFID application is used to detect whether the patient eat the pills on time or not.

When the project just started, although the NFC chip in mobile or tablet was

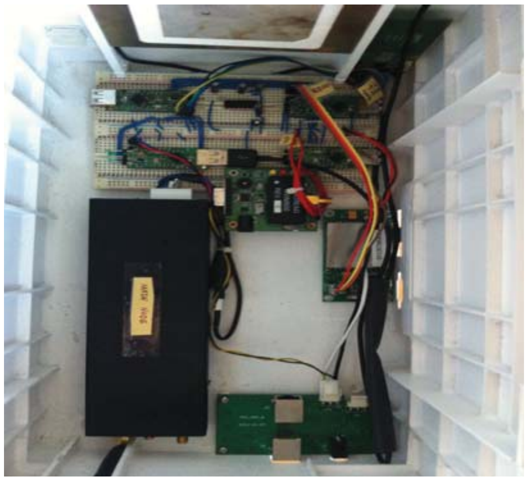


Figure 7.1. Medicine Box Demo

already supporting NFCIP-1 peer-to-peer communication, Android didn't offer software developers the P2P communication software stack or API at that time. NFC was a very new ideal which only used in mobile payment. So we handled out this new and useful ideal to use NFC into WSN system. Right now, before my thesis finished, Android already has official software stack for NFC-enable tablet peer-to-peer communication. That can helps this NFC-enable WSN system develops better in the future.

Chapter 8

Future Work

To implement a fully developed NFC-enable WSN system, there still a lot of things to do.

8.1 Support SPI and I2C Interface

In the thesis project, we only implemented sensor node connect with NFC adapter using UART interface. Actually PN532 also supports SPI and I2C interface. And Considering of SPI has higher speed than UART, it is worth to enhance the SDK to support SPI interface. This also leads another aspect can be done in the future. We need make our own PCB for the NFC apter in stead of using NXP demo board again. It will be more flexible and cheaper.

8.2 Improve and Optimize SDK Code

In addition the SDK code still need to improve and optimize in the future. Now the SDK code is just for small project, integrated is not very easy and functionality is not enough. In the future, the SDK should support NFC card reading, different P2P communication protocols and emulating a ISO/IEC 1443-4A card. And for the UART interface function, flow control should be supported to decrease the delay of communication.

8.3 communicate directly with NFC-enable tablet

Furthermore only one NFC adapter will be involved in the NFC-enable WSN system. This NFC adapter is used to connect with sensor node. And for the NFC-enable tablet, there will be no need for a additional NFC adapter. NFC peer-to-peer communication will be directly between a NFC apter connecting sensor node and a NFC-enable tablet.

Reference

[1] *iPack Center homepage.* [Online]. Available: http://www.kth.se/en/ict/forskning/centra/ipack/

[2] Z. Z. Q. C. L. Z. Zhibo Pang, Jun Chen, "Global fresh food tracking service enable by wide area wireless sensor network."

[3] N. Forum, "The keys to truly interoperable commmunications," in *NFC Forum Marketing White Paper*, 2007.

[4] G. P. Charl A. Opperman, "A generic nfc-enabled mearurement system for remote monitoring and control of client-side equipment," in *2011 Third International Workshop on Near Field Communication*, 2011.

[5] *NFC Froum homepage.* [Online]. Available: www.nfc-forum.org

[6] D.Baddeley, "Iso/iec fcd 14443-3," Jane 1999.

[7] B. Jiang, "Nfc vs iso14443 vs felica."

[8] E. International, "Near field communication interface and protocol (nfcip-1)."

[9] NXP, "Pn532 application note," in *AN104491*, December 2006.

[10] F. electronics Egypt Ltd., "Comparison of wireless technologies(nfc-wifi-zigbee-blueetooth-gsm)," January 2012.

[11] J. P. A. Y.-o. I. K. Esko Strommer, Jouni Karrtinen, "Application of near field communication for health monitoring in daily life," in *EMBS Annual International Conference.*

[12] G. C. S. N. Jose Brave, Ramon Hervas, "Adapting technologies to model contexts: Two approaches through rfid & nfc."

[13] *gartner.* [Online]. Available: www.gartner.com/it/page.jsp?id=2237315

[14] *nfctimes.* [Online]. Available: www.nfctimes.com

[15] *chipworks.* [Online]. Available: www.chipworks.com

[16] NXP, “Near field communication (nfc) controller,” December 2011.

[17] P. Semiconductors, “80c51 family programmer’s guide and instruction set.”

[18] NXP, “Pn532c106 demoboard,” in *AN10688*, February 2008.

[19] T. Instruments, “Msp430f21x2,” in *SLAS578J*, January 2012.

[20] FTDI, “Vinculum-ii embedded dual usb host controller ic,” 2009.

[21] ——, “Vnc2 debug module datasheet,” May 2010.

[22] ——, “Vnc2-32q deveopment module datasheet,” May 2010.

[23] *libnfc.org*. [Online]. Available: www.libnfc.org/documentation/introduction

[24] NXP, “Nfc fri sdk start guide,” in *AN106641*, October 2007.

[25] ——, “Nfc fri software development kit 1.0 user manual,” September 2007.

[26] ——, “Integrationguidelines documentation,” October 2007.

[27] N. Semiconductors, “Hal sdk pn5xx v2.2 quick start guide,” November 2007.

[28] ——, “Porting nfc hal software pn5xx board,” October 2007.

[29] ——, “Pn532 user manual rev.02,” November 2007.

[30] ——, “Hal api documentation,” November 2007.

[31] M. Axelsson, “Implmentation of nfc on an arm cortex-m3 based platform,” Master’s thesis, KTH, 2011.

[32] FTDI, “Vnc2 application note and vinculum-ii tool chain getting started guide,” August 2010.